8-2016

Continuum Surrogate Software Interface for Teleoperation of Continuum Robots

Ryan Matthew Scott
Clemson University, rscott6@clemson.edu

Continuum Surrogate Software Interface for Teleoperation of Continuum Robots

---

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Masters of Science
Electrical Engineering

---

by
Ryan Matthew Scott
August 2016

Accepted by:
Dr. Ian Walker, Committee Chair
Dr. Apoorva Kapadia
Dr. Keith Green

Abstract

This thesis presents a novel teleoperation interface for continuum robots. Previous teleoperation interface methods for continuum robots did not include a natural mapping due to a degree-of-freedom mismatch, using non continuum input devices with fewer degrees-of-freedom than the robot that was being controlled. The approach introduced in this thesis involves creating a 3D model of the robot using graphics libraries and a continuum kinematic model, then manipulating that graphical 3D model on screen to directly control the continuum robot. This thesis details the development of both the model and software. The teleoperation interface was developed specifically for a nine degree-of-freedom pneumatically-driven extensible continuum robot (OctArm), but it applies to any continuum robot with an arbitrary number of sections due to its modular design. Experiments using the aforementioned system on two different continuum robots are reported and areas for future work and improvement are detailed.

Dedication

To my grandparents for always believing in me and encouraging me to do my best.

Acknowledgments

I would first like to express my deep gratitude to Dr. Ian Walker and Dr. Apoorva Kapadia for giving me the opportunity to work on this project and for all of their patience, assitance and time. It has been a privilege to work with them on this project.

Special thanks go to Dr. Keith Green for being on my committee and going through the hoops to stay on board as he transitions to Cornell. I have greatly enjoyed working with him during my time here at Clemson.

I would also like to thank Mr. Glen Peterson of the Greer Programmers Group for his programming expertise and assistance. His knowledge of Java and networks communication was instrumental to the completion of this project.

I must also acknowledge my colleagues Michael Wooten and Chase Frazelle for their assistance, support and advice.

Finally, I also would like to thank my family for all of their support, without their support I would not have finished this thesis.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Robot use has expanded greatly over the last fifty years. The vast majority of robots used in industry are rigid-link manipulators. As the term implies, the links of these robots are rigid and connected at specific joints. These joints are almost exclusively revolute (rotary) or prismatic (linear). Industrial robots are used widely for their ability to perform menial and/or dangerous tasks with high levels of accuracy and repeatability. Their accuracy and repeatability comes at the cost of requiring a structured environment with little, if any, deviation between tasks.

1.1 Continuum Robots

This thesis presents a new method of teleoperative control for continuum robots. Contrary to standard rigid-link robots, continuum robots do not have fixed joints and are formed from serially connected smooth "sections". Instead of having rigid links, each section is flexible and a continuum robot can bend at any point along the backbone of each section [14] [40]. This "built-in" flexibility allows for added adaptability when it comes to grasping objects of variable size. This design is naturally inspired by elephant trunks [15][17] and octopus arms [16] [26]. Small scale continuum robots have also proven to be very helpful in minimally invasive surgery [21] [39].

The kinematics of a continuum robot are typically based on the constant curvature model which assumes that each section bends with constant curvature [12] [13] [24]. In this model, each section can be characterized by three parameters s (arc length), κ (curvature), and ϕ (orientation) shown in Figure 1.1. These three parameters are used as inputs to the kinematic model described

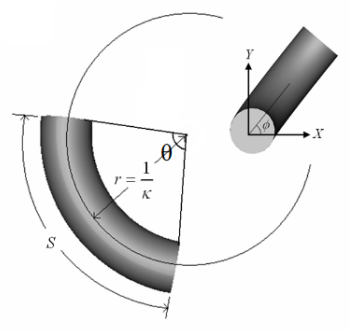


Figure 1.1: Continuum Robot Parameters [24]

Figure 1.2: OctArm and Tendril Continuum Robotic Systems

in Chapter 2 to determine the position and orientation of the OctArm [26] and tendril [42] robots (shown to the left and right respectively in Figure 1.2) under a new and novel teleoperation approach.

1.2 Methods of Teleoperation for Continuum Robots

Teleoperation is the process of a human remotely operating a robot. Robots are often used to replace humans in a task that is either mundane, too repetitive, or too dangerous for humans to complete. Teleoperation allows a user to control the robot while remaining out of harms way for those dangerous tasks that still require human supervision [37]. Although there is extensive literature on the teleoperation of rigid-link robots [28], to date very little work has been done on the teleoperation of continuum robots. In the literature, the first teleoperative interface for continuum robots mapped a joystick to a continuum robot using several different mappings [5]. Following this *Frazelle <et al >* used a rigid link robot as an input device, manipulated by a human operator to control a continuum robot [11]. Both of these however, fail to provide a natural mapping since each input device has fewer degrees-of-freedom than the continuum robot being controlled and both are based on kinematically dissimilar input devices than the continuum robot. The degree-of-freedom mismatch complicates the controls making the approaches [5], [11] difficult to learn as well as making some input motions very difficult to mimic with the robot. This thesis presents the first continuum-to-continuum interface for teleoperative control. An intuitive natural mapping is synthesized and implemented, which is easier to use and provides for a better user experience.

1.3 Thesis Overview

The continuum surrogate software, which forms the core of the method introduced in this thesis, follows the client-server model (depicted in Figure 1.3) and is thus split into two parts: the client and the server. The server renders the user interface and sends data to the client. After receiving robot parameters from the server, the client interprets the data and then sends the appropriate signals to the hardware to move the physical robot. Chapters 2 and 3 cover the server and client software design process, respectively. (If the reader is unfamiliar with either of these languages it is recommended they look at Appendix A, Glossary of Programming Terms). Chapter 4 describes the results of several validation experiments run on two robots, the OctArm and

the tendril, to verify the effectiveness of the approach as well as details regarding the hardware of both. Chapter 5 summarizes the results, draws conclusions, and lists some suggestions for future work. The entirety of the code can be found in Appendices B and C and is also available online at https://github.com/rymasc/x3d-continuum-interface.

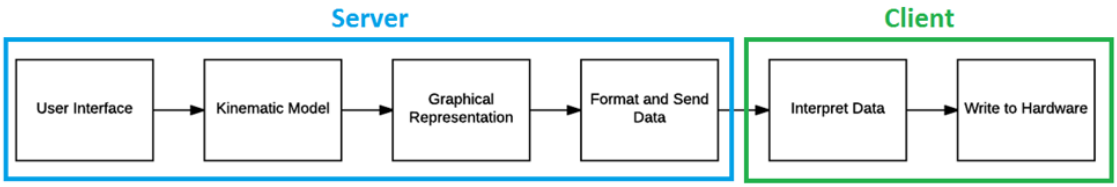


Figure 1.3: Server Client Model

Chapter 2

Server Software Design

This chapter describes the server code. In short, the server loads the user interface, runs necessary code to render the graphics model of the robot and sends the appropriate data to the client which controls the actual robot. In this chapter the server code is broken up into six sections - Model Design, JavaScript Objects, User Interface, JavaScript Libraries, Kinematic Model and Data Transmission. Each section includes purpose, evaluation of potential solutions, implementation and functionality.

2.1 Model Design

The 3D model is the core of the program so determining the graphics package for the model was the first priority. The selection criteria of the graphics package included use of known programming langauges to minimize complexity, ease of use and cost, the ability to mimic bending continuum section motions and ease of integration. Three options were considered:

2.1.1 Unity 3D

At the time of publication of this thesis, Unity 3D is the most popular game engine, making up 47% of the market [38]; Unity 3D is offered in a free personal edition. Unity 3D was the first consideration for 3D modeling because it has great documentation and support. The OctArm structure could have been modeled by creating three cylinders and then attaching them to each other. Each cylinder would represent each of the three sections and they would have to be able to

bend like each robot section. However, primitive cylinders in Unity only have vertices on the top and bottom allowing only for shearing but not bending. An attempt to solve this issue was made by modeling cylinders that had vertices distributed along the length with Maya, a 3D modeling software. However when the cylinders were imported into Unity there was no clear way to manipulate the vertices. For this reason Unity was abandoned as the modeling technology for this project.

2.1.2 VRML, Open Inventor and Coin 3D

The next graphics package to evaluate was determined by looking at some of the simulation modeling done by *Jones and Walker* [23]. Previous modeling of the OctArm and AirOctor robots used Virtual Reality Modeling Language (VRML), and some supporting libraries, Open Inventor and Coin 3D [23]. After looking into VRML it became apparent that most of the old code was now deprecated[1] [3].

```
<X3D>
    <Scene>
        <Shape>
            <Appearance>
                <Material diffuseColor="1 0 0"></Material>
            </Appearance>
        <Box></Box>
        </Shape>
    </Scene>
</X3D>
```

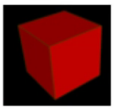


Figure 2.1: X3D Box Node

2.1.3 X3D

X3D is the successor of VRML. X3D (eXtensible 3D) is a file format used to represent 3D scenes and objects released by the Web3D Consortium [3]. X3D expands upon VRML and adopts an XML syntax, which makes the language quick and easy to learn. X3D nodes are XML elements, a set of opening and closing tags and are identified with the .x3d extension. Every X3D file has at least one <`Scene`> node which contains all the viewpoints, shapes and associated transform nodes. In X3D, a node is defined as a set of XML tags. Figure 2.1 shows a simple X3D file that defines a red cube using the box node [1]. The `diffuseColor` attribute is set to the string `"1 0 0"` to make the cube red. Here the three numbers correspond to red, green and blue values, where the max value for a given color is `1`. This simple box node example uses default parameters – one cubic

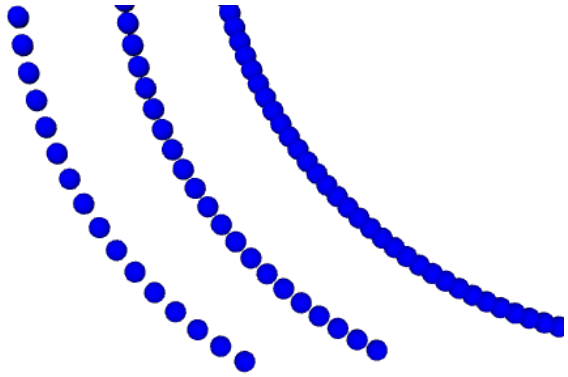


Figure 2.2: Approximation of Bending Cylinder with Spheres

meter centered at the origin. The position and orientation can be changed if the <`Shape`> node is wrapped in a <`Transform`> node with the desired translation and rotation properties set. X3D primitives do not have enough vertices to allow for bending. Some of the models found in [20] gave us the idea to use stacked spheres to approximate cylinders as shown in Figure 2.2. Using this idea, the sphere nodes are moved individually based on the kinematic model (Section 2.5) to model the bending and extension needed to accurately depict a continuum robot section. X3D was chosen due to the simple XML syntax and ability to use JavaScript [6] to dynamically modify the Sphere nodes.

```
<transform id="b0" DEF="b0" translation='0 0 0.0070'>
    <shape>
        <appearance>
            <material diffuseColor='0 0 1'></material>
        </appearance>

        <sphere radius="0.0174"></sphere>
    </shape>
</transform>
```

Figure 2.3: Model X3D Sphere Format

Comparing the simple syntax that defined an X3D Box in Figure 2.1 to that of Figure 2.3, the latter wraps the sphere in a transform node with a unique id and also defines a radius. Each section is ultimately defined by 45 spheres. The format of the id is *b0-b44*, *m0-m44*, and *t0-44*, for a total of 135 spheres in the robot, where b represents the base, m the middle and t the tip sections respectively.

2.2 JavaScript Objects

JavaScript is an Object-Oriented Programming (OOP) language. In an object-oriented programming language code is based on organizing data instead of logic. Using an OOP language allows for the creation of modules, which helps to create repurposeable code. The OctArm model is defined by five object files, or modules: `Octarm.js`, `Section.js`, `x3dSphere.js`, `Color.js` and `Coordinate.js`. The code has the following hierarchy, `Octarm.js` contains three instances of `Section` for the OctArm base, mid, and tip sections. The `Section` object contains an array of `x3dSphere` objects that each have an associated `Color` and `Coordinate`. Since the model of the robot is created by using a series of spheres, this object oriented programming approach works well for this project.

`Octarm.js` has four instance variables (`anchor`, `base`, `mid` and `tip`), one member variable (`n`, which represents the number of spheres in each section), and three functions (curve, orient and extend). Each one of the sections is formed by making an array variable, `objs[]`, that reads all the appropriate (*b0-b44*, *m0-m44* or *t0-t44*) tags out of the model. In addition to the instance variable array there is the anchor variable which corresponds to the last `x3dSphere` in the previous section, or in the case of the base, the anchor. There is also a `type` variable and then variables for `radius` and `color`. The `type` variable stores the section type (base, midsection or tip) and the `radius` and `color` variables stores the size of the radius and the color (red, green or blue) of the spheres. Finally, there are the robot parameter variables `s`, `k`, `phi`, and `d` which are defined in the kinematics section (Section 2.5). `Color` and `Coordinate` are helper files that translate the strings into three member variables. In the case of `Color` they are `r`, `g`, `b` (for red, green blue) and in `Coordinate` `x`, `y`,`z`.

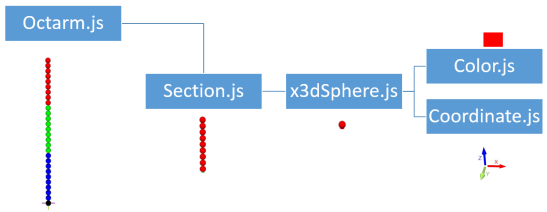


Figure 2.4: JavaScript Object Hierarchy

2.3 User Interface

The 3D model is embedded inside a webpage (a single HTML file), instead of being saved with the usual .x3d extension which allows for JavaScript to access the model by selecting elements. There is a single HTML file that defines the user interface, `model.html`. The `model.html` file contains essentially two components — the controls section and the X3D section. The controls section includes check boxes, button commands and information regarding the parameters of each section of the OctArm and the X3D section places the X3D content in an X3D element, `<x3d></x3d>`. This HTML file is then saved with a .ejs extension in the views folder so that it can be rendered with the node server. Despite the fact that the ejs file is viewed using a web browser that acts as a client, all the underlying JavaScript that controls the user interface actually runs as a server.

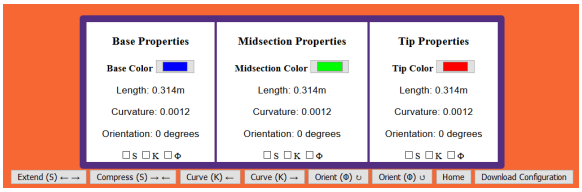


Figure 2.5: Control UI

The User Interface (UI) features a feedback segment that allows the user to see the length (s), curvature (κ) and orientation (ϕ) for a given section, or multiple sections. It also has check boxes that allow the user to select the parameter they wish to modify for the given section. There is one large row with buttons to call one of the three main OctArm functions - extending, curving, and orienting. These functions increment s, κ and ϕ respectively. For each movement function at least one section must be selected and then the appropriate button increments or decrements that value for the selected section(s), applies it to the model and sends the data to the client program. Note, that it is not valid to select two spatially distinct sections because there is no intuitive reason to move them in unison. The home button simply reloads the page and resets everything and Download Configuration allows the user to save a .txt file that contains the current value of the robot parameter variables in the format of a comma separated value string.

The X3D Section contains X3D spheres in the format shown in Figure 2.6 but also contains three cylinders and a ViewPoint node all wrapped in Transform nodes. The ViewPoint node is arranged so that entire model is viewable at an appropriate zoom level. The three cylinders are

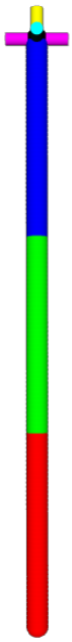


Figure 2.6: X3D Octarm Window

placed at the origin and are rotated ninety degrees apart from each other to represent the X, Y and Z axes as a triad. Lastly, there is one extra black sphere called the anchor node. Each section is tied to the previous one; tip is tied to the last sphere in the midsection which is tied to the last sphere in the base and the base is tied to the anchor. As far as controlling the view, scrolling the mouse wheel zooms in and out. Holding the left mouse button down and moving the mouse rotates the view while holding the middle mouse button down and moving the mouse pans. Doing the same action with the right mouse button results in an additional way to zoom in and out.

2.4 JavaScript Libraries

With the model defined in terms of the graphics package and sectioned off into JavaScript objects, the next step is to use libraries to enable movement of the 3D model in response to the user interface. X3DOM [10] is the most critical library in the project. The name actually is a portmanteau of X3D and DOM, where DOM refers to the Document Object Model, the basis API for creating dynamic web pages with JavaScript. This library is what enables regular JavaScript code to modify the translation attributes, and anything else in the X3D model, to move the spheres.

X3DOM is developed by the Fraunhofer Institute and is intended to become the official standard for declarative 3D content in HTML5 [10].

jQuery is a JavaScript library that makes it easier to access HTML elements [25]. The JavaScript code for setting a text fields value to "text" versus the jQuery method are shown in Figure 2.7. Clearly, jQuery has a simpler syntax which makes for shorter code and quicker debugging.

```
JavaScript    document.getElementById("id").innerHTML = "text";
jQuery        $("#id").text("text");
```

Figure 2.7: JavaScript vs jQuery

There are a few other small libraries in addition to X3DOM and jQuery. Sylvester is a vector and matrix library written by James Coglan that simplifies working with matrices [2]. The `transform.js` library contains all of the kinematic equations. The remaining libraries were developed in house; `qwikMath.js` includes some basic mathematical formulae and `fileComm.js` contains all the code needed to send data to the client.

2.5 Kinematic Model

A kinematic model describes the geometric pose or motion of an object without considering its mass or forces. Forward kinematics is the process of finding the end-effector position and orientation in terms of the robots configuration variables. The configuration variables of a rigid link robot are either angles for revolute joints or lengths for prismatic joints. Solving the forward kinematics problem for the robot requires defining a frame for each joint with the base frame set at the robot base and the tool frame set at the faceplate of the end-effector. Each frame can be defined in terms of another frame with the homogeneous transformation matrix defined in Equation 2.1 [36]. Here H_n^0 represents the transformation from frame 0 to frame n.

$$H_n^0 = \begin{bmatrix} R_n^0 & d_n^0 \\ 0 & 1 \end{bmatrix} \tag{2.1}$$

where R_n^0 is the 3×3 rotation matrix representing the change in orientation between the two frames and d_n^0 is the offset 3×1 vector representing the translational offset vector between frames 0 and n.

One of the most commonly used methods used to generate this formulation is the Denavit-Hartenberg Method. Typically defining a rigid body transformation requires six parameters. The Denavit-Hartenberg technique makes use of careful frame coordinate selection to reduce the requirement to four parameters [36]. These parameters are a_i (link length), α_i (link twist), d_i (link offset), and θ_i (joint angle) between frame $i-1$ and frame i. They are combined within a special homogeneous transformation matrix defined in Equation 2.3.

$$H_i = Rot_{z,\theta_i} \times Trans_{z,d_i} \times Trans_{x,a_i} \times Rot_{x,\alpha_i} \tag{2.2}$$

$$H_i = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

where $c_x = \cos(x)$, $s_x = \sin(x)$ and i represents the i^{th} transformation between frame $i-1$ and frame i. Equation 2.4 is obtained from taking the matrix product of $[H_1^0]...[H_n^{n-1}]$ where n is the number of joints. Three out of the four parameters are constants and the fourth is a variable. The values of these parameters are typically encoded in a link parameter table. (See Table 2.1 for an example of this).

Hannan and Walker found kinematics of continuum robots operating in the plane can be found via a "virtual" rigid-link RPR manipulator [40][18]. Here the continuum section in the plane can be represented by a rotation of θ, translation by $||x||$ and a rotation by θ. From here the link parameter table is generated below (Table 2.1):

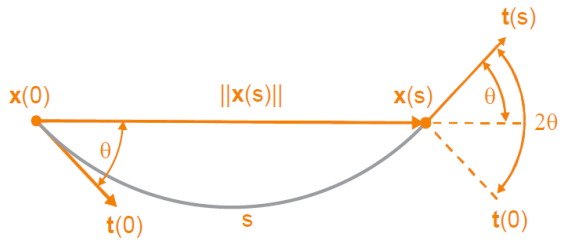


Figure 2.8: Continuum Robot Section represented as an RPR Manipulator

Table 2.1: Link Parameter Table

Link	θ	d	a	α
1	*	0	0	-90
2	0	*	0	90
3	*	0	0	0

Substituting the link parameter values into Equation 2.3 and multiplying the resulting matrices together yields:

$$H_3^0 = \begin{bmatrix} c_{\theta_1+\theta_3} & -s_{\theta_1+\theta_3} & 0 & -d_2 c_{\theta_1} \\ s_{\theta_1+\theta_3} & c_{\theta_1+\theta_3} & 0 & d_2 s_{\theta_1} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

From here θ_1, d_2 and θ_3 have to be converted to s, κ and ϕ. Observing Figure 2.8, the following relationship can be seen:

$$\theta_1 = \theta_3 = \theta \tag{2.5}$$

$$d_2 = ||x(s)|| \tag{2.6}$$

Using $s = r(2\theta)$ and Equation 2.5:

$$s = r(\theta_1 + \theta_3) = (\theta_1 + \theta_3)/k$$

$$(\theta_1 + \theta_3) = sk \tag{2.7}$$

Using Equation 2.6:

$$\frac{||x(s)||}{2} = \frac{d_2}{2} = r\sin\theta = \frac{\sin\theta}{\kappa}$$

$$d_2 = \frac{2\sin\theta}{\kappa} \tag{2.8}$$

Substituting Equations 2.7 and 2.8 into the Denavit-Hartenberg general RPR form (2.4) results in (2.9).

$$\begin{bmatrix} c_{s\kappa} & -s_{s\kappa} & 0 & \frac{c_{s\kappa}-1}{\kappa} \\ s_{s\kappa} & c_{s\kappa} & 0 & \frac{s_{s\kappa}}{\kappa} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

Extending this result to three dimensional space, and modeling the spatial section kinematics using an RPRR manipulator results in a similar result that includes ϕ (Equation 2.10)[17].

$$\begin{bmatrix} c_\phi c_{s\kappa} & -s_\phi & -c_\phi s_{s\kappa} & \frac{c_\phi - c_\phi c_{s\kappa}}{\kappa} \\ s_\phi c_{s\kappa} & c_{s\kappa} & -s_\phi s_{s\kappa} & \frac{s_\phi - s_\phi c_{s\kappa}}{\kappa} \\ s_{s\kappa} & 0 & c_{s\kappa} & \frac{s_{s\kappa}}{\kappa} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

The models (2.9) and (2.10) form the (planar and spatial, respectively) basis for the models in this thesis. However, the Denavit-Hartenberg algorithm assumes that the point of rotation is the Z-axis, but in the X3D coordinate system the point of rotation is the X-axis.

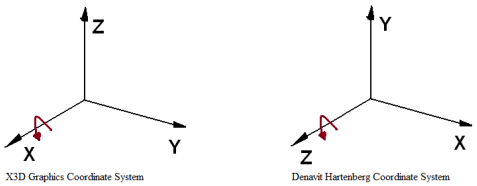


Figure 2.9: Coordinate Systems

Equation 2.9 is based on Z-rotations with the motion components in the XY-plane. To put this in a form that will work conveniently with the X3D model, several row and column swaps need to be performed. Essentially the rotational component has to be in the form of a general X-rotation matrix and the translational components need to be in the YZ-plane.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{s\kappa} & -s_{s\kappa} & \frac{c_{s\kappa}-1}{\kappa} \\ 0 & s_{s\kappa} & c_{s\kappa} & \frac{s_{s\kappa}}{\kappa} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

Following a similar technique of row and column swapping (2.10) can be transformed into (2.12).

$$\begin{bmatrix} c_\phi & -s_\phi c_{s\kappa} & s_\phi s_{s\kappa} & \frac{s_\phi - s_\phi c_{s\kappa}}{\kappa} \\ s_\phi & c_\phi c_{s\kappa} & -c_\phi s_{s\kappa} & -\frac{c_\phi - c_\phi c_{s\kappa}}{\kappa} \\ 0 & s_{s\kappa} & c_{s\kappa} & \frac{s_{s\kappa}}{\kappa} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.12}$$

Equation 2.12 represents the homogeneous transform matrix for a general section in 3D space. In the following, H_b, H_m, and H_t represent the homogeneous transform matrices for the base section, midsection and tip section respectively. The right column of the H matrix determines the (X, Y, Z) position of a given spheres' center. H_b is found for each incremental ball using appropriate s, κ and ϕ values for each individual sphere. Since a given section will have constant curvature and orientation κ and ϕ will be constants, the variable s will be represented for each sphere by the incremental arc length s_i that is defined by:

$$s_i = d \times (i + 1) \tag{2.13}$$

where d refers to the spacing between spheres and i is the sphere number. Since the section is allowed to extend; d is a variable and can be calculated by taking the given section length and dividing it by the number of spheres. The homogeneous transform matrix is a function of s_i, κ and ϕ, noted as $H(s_i, \kappa, \phi)$. Equation 2.14 is used to find the x, y, z positions of the base spheres.

$$B = H_b(s_i, \kappa, \phi) \tag{2.14}$$

$$B = [b_1\, b_2\, b_3\, b_4] \in \mathbb{R}^{4\times 4}$$

where $\times$ represents standard matrix multiplication.

$$b = b_4 \tag{2.15}$$

However, this does not extend directly for finding the positions of the spheres in the mid-section and tip. To find the positions of the spheres in these sections the previous homogeneous transform matrices must be multiplied together.

$$M = (H_b(s, \kappa, \phi) \times H_m(s_i, \kappa, \phi)) \tag{2.16}$$

$$M = [m_1\, m_2\, m_3\, m_4] \in \mathbb{R}^{4\times 4}$$

$$m = m_4 \tag{2.17}$$

$$T = (H_b(s, \kappa, \phi) \times H_m(s, \kappa, \phi) \times H_t(s_i, \kappa, \phi)) \tag{2.18}$$

$$T = [t_1\, t_2\, t_3\, t_4] \in \mathbb{R}^{4\times 4}$$

$$t = t_4 \tag{2.19}$$

where b, m, and t are 4×1 column vectors of the following form

$$\begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \tag{2.20}$$

and each homogeneous transform matrix uses s_i, κ, and ϕ for the corresponding section, i.e. H_b uses base values, H_m uses midsection values, etc.

2.6 Data Transmission

Originally the X3D JavaScript was intended to run in the web browser, but when it came time to obtain the robot parameter data there was no way to extract it from the browser environment. JavaScript is sandboxed when it runs inside the browser; security exceptions prevent it from opening web sockets and all other forms of file input/output. By running JavaScript on the Node js framework JavaScript is allowed to bypass these exceptions to directly access the hardware. Node js is a JavaScript runtime environment based on Google Chrome's V8 JavaScript Engine [8].

There are three node modules that are used on the Node js framework **ejs**, **express** and **socket-io**. Firstly, ejs (Effective JavaScript) is a templating module for Node. The express module provides functions which allow for the creation of a simple web server. Lastly, the socket-io module provides a way to open, send and receive data over a web socket.

```
io.on('connection', function(socket){

   console.log('a user connected');
   socket.on('disconnect', function(){
      console.log('user disconnected');
   });

   socket.on('message', function(msg){
      console.log("Wrote JSON\n");
      io.emit('message', msg);
   });
});
```

Figure 2.10: Server.js Event Handlers

```
formatData : function formatData(robot) {
   var obj = {
   "params":
      [
         {"base_s" : robot.base.length},
         {"base_k" : robot.base.k},
         {"base_phi" : robot.base.phi},
         {"mid_s" : robot.mid.length},
         {"mid_k" : robot.mid.k},
         {"mid_phi" : robot.mid.phi},
         {"tip_s" : robot.tip.length},
         {"tip_k" : robot.tip.k},
         {"tip_phi" : robot.tip.phi}
      ]
   };
   return JSON.stringify(obj);
}
```

Figure 2.11: JSON Format

`Server.js` contains the code to create the web server with a designated IP address, render the interface and begin listening for client connections. To render the interface express utilizes two folders: a public folder that holds all the cascading style sheets (CSS) and JavaScript and the views folder which holds the ejs file. The socket-io module creates two event handlers — one runs when the server gets a connection and the other runs when the server gets a message. These event handlers are shown in Figure 2.10. When the server gets a message it sends that message over the port to the Java client code using the `socket.emit()` function — either `OctarmClient.java` or `TendrilClient.java` depending on which robot it is controlling.

When a button in the UI is clicked, it calls the appropriate block of code to modify the

s, κ, ϕ parameters, move the model, format the data and send it to the server. This is done with the function call `fileComm.sendData(fileComm.formatData(robot));` from `main.js`. The `formatData(robot)` function, nested in the `sendData()` function, returns the JSON data structure. JSON or JavaScript Object Notation is a popular data-interchange format [7]. The format can either be an unordered set of key value pairs, an array of values, or a combination of the two. The robot parameters are stored in one key-value pair where the key is `"params"` and the value is an array of nine key-value pairs as shown in Figure 2.11. Originally the JSON format was simply a set of nine key-value pairs, but since they are unordered it makes more sense to put them in an array that preserves order so they can be read out directly on the client side.

2.7 Server Program Flow

The JavaScript code is contained three folders: **node_modules**, **public** and **views**. The **views** folder holds the ejs file to be rendered and **node_modules** holds all the necessary node js libraries. **Public** contains all the core JavaScript and CSS that control the interface and model. The JavaScript is split into two main categories: **libraries** and **objects**. The **libraries** are files that contain groups of related functions while the **objects** are definitions for various parts of the robot.

`Main.js` is the entry point for this program, it acts as an event handler for the web page. Once the document model loads it immediately creates an instance of `Octarm.js` which in turn creates three instances of `Section.js` and 136 instances of `x3dSphere.js` — 45 of each section plus the anchor node. It does this by creating the robot object which is an instance of the `Octarm.js` file, in the scope of `Main.js`. The curve, extend and orient functions are increment the appropriate values. Afterwords, the transform kinematics function described in Section 2.5 makes all the necessary updates to move the spheres. From there it calls the `fileComm.js` library and writes to the web socket. This process is shown in Figure 2.12.

Figure 2.12: JavaScript Client Program Flow

Chapter 3

Client Software Design

The client software is responsible for reading a message from the server, interpreting the message and then writing the corresponding values to the hardware to physically move the robot. There are two different versions of the client code — one for the OctArm robot and another for the tendril robot; both of these are written in Java [19] using Intellij IDEA [22] as the Integrated Development Environment (IDE). Both of these versions are identical in how they receive and parse data; they differ in how they interpret data and then write that interpreted data to hardware. The Java code uses the socket-io java client API allowing for simple communication between JavaScript and Java. The Java program has an event handler that calls a function every time it receives data. The data is received in JSON format and converted into float values. This process is outlined in the Java Socket-IO section. From this point on the code is unique to the robot being controlled.

3.1 Java Socket-IO

The Java code uses Maven [9], which is a software build tool that manages dependencies, following the Project Object Model (POM). The POM sets a projects configuration with information including the projects name, owner and a list of dependencies which is all contained in the `pom.xml` file. Socket-IO is one of the dependencies for this project.

On startup of the Java code, the `OctarmClient` or `TendrilClient` constructor calls the `setupConnection()` function, which connects to the socket using the `IO.socket("hostname:port")` function. Then the constructor defines event handlers for connecting to the socket, disconnecting

```
socket.on(Socket.EVENT_CONNECT, new Emitter.Listener() {
   public void call(Object... objects) {
      System.out.println("Socket Connected");

   }
});
```

Figure 3.1: Socket Connection Event Handler

```
            try{
               JSONObject jsonData = new JSONObject(objects[0].toString());
               JSONArray paramArray = jsonData.getJSONArray("params");
               float[] paramData = new float[9];
              if(paramArray !=null){
                 for (int i = 0; i < paramArray.length(); i++) {
                    JSONObject param = paramArray.getJSONObject(i);
                    paramData[i] =
                        Float.parseFloat(param.getString(JSONObject.getNames(param)[0]));
                 }
                 //Write to Tendril or Octarm
              }

            } catch (JSONException e){
                e.printStackTrace();
             }
```

Figure 3.2: Translating JSON to Float Array

from the socket, receiving an error from the socket, and receiving a message from the socket using the `socket.on()` function. The general format of an event handler is shown in Figure 3.1. The first argument is the event type and the second argument is the function to be called. This function, `call()`, is defined inside the `Emitter.Listener()` object. In the example, the event handler simply prints out a message to the user that the socket connected. In both the OctArm program and the tendril program the first process is to convert the JSON message into an array of floats. This is done by stringifying (a built in JSON Method) the `objects` argument from the `call()` function into a `JSONObject`. Since the JSON Data is in the form of an array, shown in Figure 2.11, the `JSONArray` is extracted by using the `"params"` key. From there the float array is built by reading out each element and casting it to a float from a string.

3.2 OctArm Client

All the OctArm client needs to do is receive the s, κ, ϕ parameters, convert them to voltages and then write them to the hardware. Before the completion of this project Simulink was used exclusively in the lab to communicate with the Quanser boards. The preexisting Simulink model took s, κ and ϕ values as inputs and then converted them into voltages to drive the pressure regulators. For the first attempts to control the OctArm the goal was to use Matlab, since that code was already written.

Various communication interfaces were evaluated after it was clear that there was no way to directly communicate between JavaScript and Matlab. TCP/IP was rejected because there is no TCP/IP Client Block available in Simulink. A custom block could not be created because TCP/IP Matlab connections would only work if Matlab was the server. At this point, the project forked where a C# - ASP.NET interface was examined for the OctArm while socket-io was being considered for the tendril. C# - ASP.NET was evaluated for the OctArm because code existed to communicate between C# and Simulink in the lab. Similarly, code had already been written to communicate between Java and Arduino. The Java Socket-IO ended up working first, so the next step was to determine how to communicate with the Quanser board using Java.

Quanser has a built in Java Communications API. This API allows the user to communicate between Java and Quanser Simulink Blocks. However, if the API is used it requires the use of special Java Communications Quanser Blocks, necessitating a rewrite of the Simulink model, since this API has never been used before. Since Matlab offered no advantages at this point, it was removed from the loop. The preexisting Matlab code to convert the configuration variables s, κ, ϕ to voltage was easily translated into Java. The only change required was to declare each variable in Matlab as a `float` because Java is statically typed. To assign a floats in Java, numbers must be appended with an `f` literal, i.e. `2.4f`, to distinguish between the type `double`. At this point, the only missing functionality was the ability to connect to hardware directly with the Java program. However, there is no way to directly connect to the hardware with Java, because Quanser only released a HIL (Hardware-In-the-Loop) API for C. This problem was solved with the Java Native Interface, which allows for Java to directly communicate with native C code.

3.3 Tendril Client

The Tendril Client has two classes, `TendrilClient` and `TendrilSystem`. The Socket-IO library is used in the `TendrilClient` class. In addition to the event handler definitions described in section 3.1, there is a function to set up the serial port and a listener class to read from the serial port. This function runs immediately on startup and the listener class `PortReader` is bound to the port. It also instantiates a `TendrilSystem` object which manages the conversion of the s, κ, ϕ configuration variables into meaningful outputs to control the tendril.

The tendril hardware is described in more detail in Chapter 4. The tendril has three sections each controlled by three tendons. These tendons are equally spread around the section 120° apart. Each set of motors is offset 40° from the previous one. The base section is controlled by motor 0, 1 and 2 (at 0°, 120°, 240°), the midsection is controlled by motor 3, 4 and 5 (at 40°, 160°, 280°), and the distal section is controlled by motor 6, 7 and 8 (at 80°, 200°, 320°). These values are stored in an array called `motorLocations` where the index represents the motor number and the value represents its angle. The tendril has not been modeled kinematically as with the OctArm - so unlike the OctArm there was no preexisting code to convert between s, κ, ϕ to output voltages. This conversion process takes place in the `TendrilSystem` class. All of the s values were left out in this parameterization. The class only uses six parameters to define the robot — the κ and ϕ for each section. Each section's curvature is converted into a percentage of maximum tension by dividing by the maximum curvature for each section of the tendril using the `sectionTensionPercentage()` function. From there, the value is rounded and converted to a byte using `convertToByte()`. In this model, ϕ values are used to determine which motors to use in the `findBaseTensions()`, `findMidTensions()` and `findDistalTensions()` functions. The tension found in the `sectionTensionPercentage()` is split between the motors as a function of distance between them, i.e. if ϕ is halfway between motor 0 and motor 1, half of the tension is applied to motor 0 and half to motor 1. Finally, there is a `changedSystemStatus()` function that monitors the state of the tension and ϕ values. If the values change by 15% or more then `TendrilClient` will write the current configuration to the arduino. This is done because the tendril control system is not able to handle rapid continuous changes in the tension value.

```
public class Quanser{
   static{
      System.loadLibrary("quanser_jni_hil");
   }
   public native void connectWrite(float[] voltage);
}
```

3.4 Java Native Interface and C Dynamic Link Library (DLL)

The Java Native Interface(JNI) allows Java code to call and be called by native code [29]. Native code is compiled to run on a specific platform and is typically written in C, C++ or assembly. To use JNI two things must be created — a JNI wrapper class and a C dll. The wrapper class consists of two parts. First it needs to load the dll using the name of the dll, in this case `quanser_jni_hil.dll` as the input to the `System.loadLibrary()` method. After that, all of the functions that need to be called from Java have to be declared using the `native` keyword. For this application we have only have one function called `connectWrite()` which will connect to the Quanser boards, write an array of voltages and then disconnect from the board, so there is only one function declaration in the wrapper class.

On Windows, a C program can be compiled into three different files — an application (.exe), a static library (.lib), and a dynamic link library (.dll). An application is a program that can run by itself. The other two are library files that cannot run on their own. A lib file is a compiled library that is added by the linker. This speeds up compiling time because the library doesn't need to be compiled each time a change is made in the program. Dynamic Link libraries (.dll) are also compiled libraries, but instead of being added by the linker they are loaded into memory at run time [27]. As a result, applications that include a lib file compile into larger executable files than an application that uses a dll.

The C code was written with Visual Studio [4]. First, all Quanser references needed be added to the Visual Studio Project. Header files were included by adding the `\$(HIL_DIR)\include` path to the project. Library files were added in a similar way with the `\$(HIL_DIR)\lib\windows` path. Here `\$(HIL_DIR)` is an Environmental Variable that points to the location where the HIL API is installed. Lastly, this project also used three lib files: `hil.lib`, `quanser_runtime.lib` and `quanser_common.lib`. To test the functionality, a simple application was written to connect to the boards, write a voltage, and disconnect. There are three main API functions used to create the C

```
\\connect function
t_error boardResult = hil_open("q8_usb", "0", &board);

\\write functions
write_result = hil_write_analog(board, channels, ARRAY_LENGTH(channels), buffer);

\\close function
hil_close(board);
```

Figure 3.3: HIL API C Functions

```
//---NORMAL C FUNCTION DEFINITION
void connectWrite(float[] voltage){
   //function definition
}
//--------------------------------
// C WITH JNI FUNCTION DEFINITION

JNIEXPORT void JNICALL Java_Quanser_connectWrite(JNIEnv *env, jobject thisObj,
    jFloatArray voltage){
   //function Definition
}
```

Figure 3.4: Changing C Function Definition to JNI Function Definition

dll file: a connect, write and disconnect function shown in Figure 3.3. The connection function is `hil_open()` which takes three inputs: the board type, board identifier and a reference to the board variable. The board is a Quanser Q8 connected via usb so the first argument is `"q8_usb"`. The board identifier is a label to the board — the first board is assigned `"0"` and the second board is assigned `"1"`. The two board variables are called `board` and `miniBoard` in the code. The first board has eight analog connections and the second board only has one analog connection, with the combination enabling the nine degrees-of-freedom of the OctArm. The write function has four inputs: the board designation, the channels' designations, the number of channels designated, and the values to write. For the first board the channels variable is an array of all the analog outputs, i.e. `channel = {0,1,2,3,4,5,6,7}`. Similarly `minichannel = {0}`.

Once tested, the next step was to configure the project to work with JNI. To do this, the project configuration type was changed from an application to a dynamic library and the `<jni.h>` header file was added from the `%JAVA_HOME%` path. The code no longer needed a `main()` function because it is not a standalone program — its entry point is defined by Java.

```
jfloat *inputArray = (*env)->GetFloatArrayElements(env, inputArray, 0);
jsize length = (*env)->GetArrayLength(env, inputArray);
```

Figure 3.5: JNI Array

In addition to the changes in project configuration, function definitions are also different using JNI as shown in Figure 3.4. The arguments `JNIEnv*` and `jobject` must always be included and listed first in JNI. `JNIEnv*` is a reference to the JNI Environment so that all JNI functions can be used. The second argument, `jobject` is a Java Object. `JNIEXPORT` and `JNICALL` are macros that are not used but are required to be there. In our wrapper class we declared a native function that took type `float[]` as an input. To include that in the definition we use type `jFloatArray`. This pattern persists in JNI for every primitive data type, there is a "j-variant" i.e. jbyte, jchar, jint, jboolean, etc. This is necessary because different compilers use different number of bits for certain data types across languages. This is done to prevent data corruption in primitive data types. JNI Arrays add one more step to this because of a difference in the way Java stores array variables. In C, arrays simply store the elements in one large block of memory. Java arrays are similar to classes and store additional data — such as the size. To use a JNI array it is essential to strip this extra information out to get only the required values. This is done using the `JNIEnv*` variable as shown in Figure 3.5.

Chapter 4

Experimental Validation

The interface described in Chapters 2 and 3 was validated on two continuum robots of different types: the OctArm and the tendril. In this chapter, the hardware setup as well as the validation of the interface for each robot is detailed.

4.1 Octarm Hardware

The OctArm is constructed with three continuum sections — a base, midsection and tip which makes a total of nine controlled variables to define the robot shape: s, κ, ϕ for each section. The OctArm is actuated with pneumatic artificial muscles known as McKibben Actuators. Each "muscle" is constructed of latex tubing covered with a plastic mesh sheathing [26]. When filled with pressurized air, the tubing expands. The sheathing prevents the tubing from expanding axially; instead allowing the tube to fully expand along its length. These muscles are grouped together in parallel to form sections. The base and midsection each have six muscles (three sets of two at the same pressure) while the tip has three. In the proximal sections the tubes act in pairs, so in each section there are three muscle groups 120°apart. Different combinations of pressures allows for complete movement of the section.

The muscles are driven by nine pressure regulators where each section is controlled by three. These pressure regulators are connected to an air compressor and two Quanser Q8-USB control boards (shown in Figure 4.1). Each board has eight analog inputs, eight analog outputs, as well as eight encoder inputs. For control of the OctArm nine analog outputs are needed, so one board

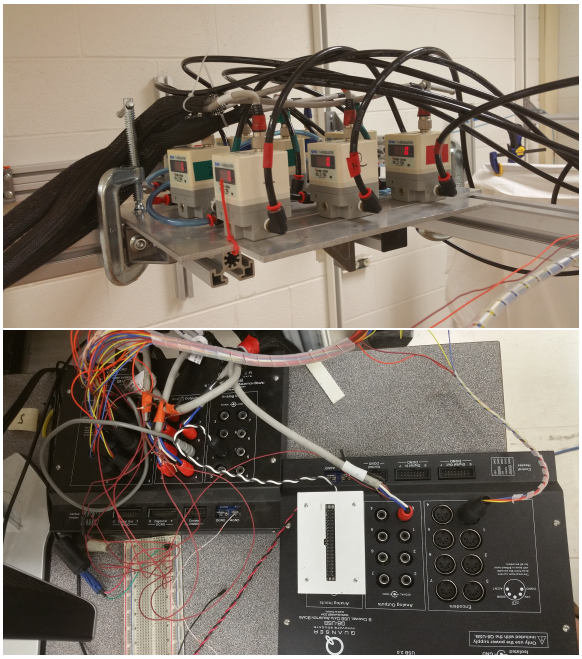


Figure 4.1: Pressure Regulators and Quanser Boards

has eight analog connections and the other has one. The OctArm also uses nine string encoders to measure the lengths of each muscle group, however these are not used in the interface used in the work presented in this thesis.

This thesis research was focused on the OctArm from the beginning so it was expected that the OctArm would perform better than the tendril. The preexisting OctArm hardware acts as a blackbox that takes s, κ, ϕ values as inputs and generates pressure regulator input voltages as outputs. The tendril control system is not as developed and a unique mapping had to be created. Four validation experiments were run on the OctArm: each section individually plus one with all sections together. General motion experiments were run on the tendril to demonstrate the interface could intuitively control the robot, however without much regard to setting up the experiment in a way to achieve high accuracy.

4.2 Octarm Validation

The purpose of the single section validation experiments was to verify the interface was correctly controlling the robot as well as to measure the accuracy of the system. To properly validate

a section, the following script was performed on a given section; extend, compress, curve right, extend, compress, orient clockwise and then return to the starting home position. Six representative snapshots of the X3D model and the OctArm are shown for the base, midsection and tip in Figures 4.2, 4.3 and 4.4 respectively. The full motion videos for the base [32], mid [33], and tip [34] are on YouTube. The interface was quick and easy to use and was proved accurate modulo the information in the kinematic model. Gravity pulls the OctArm down, the effect of which is not added in the kinematic model (see images at the top of Figure 4.2 for example). This effect is most evident when moving the base section since the base has the added weight of the midsection and the tip. The effect of not explicitly modeling gravity on the tip produced relatively little error due to the fact that the tip only needs to support itself. However, as the curvature increases these differences become more pronounced (see Figure 4.5).

After verifying that the interface worked well with single sections, the same procedure was run on all sections simultaneously. The results of this validation experiment are shown in Figure 4.5. The same gravitational effects were very easy to observe in this experiment.

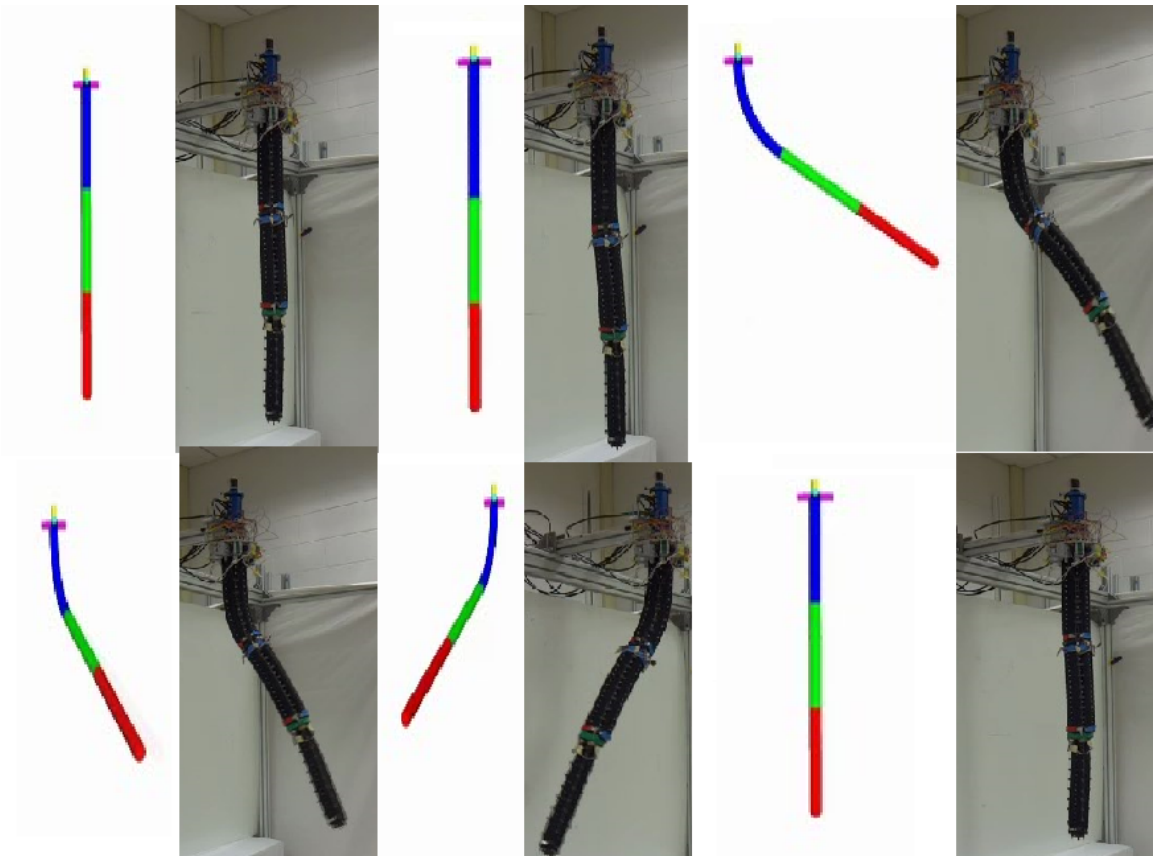


Figure 4.2: OctArm Validation of Base Section [32]

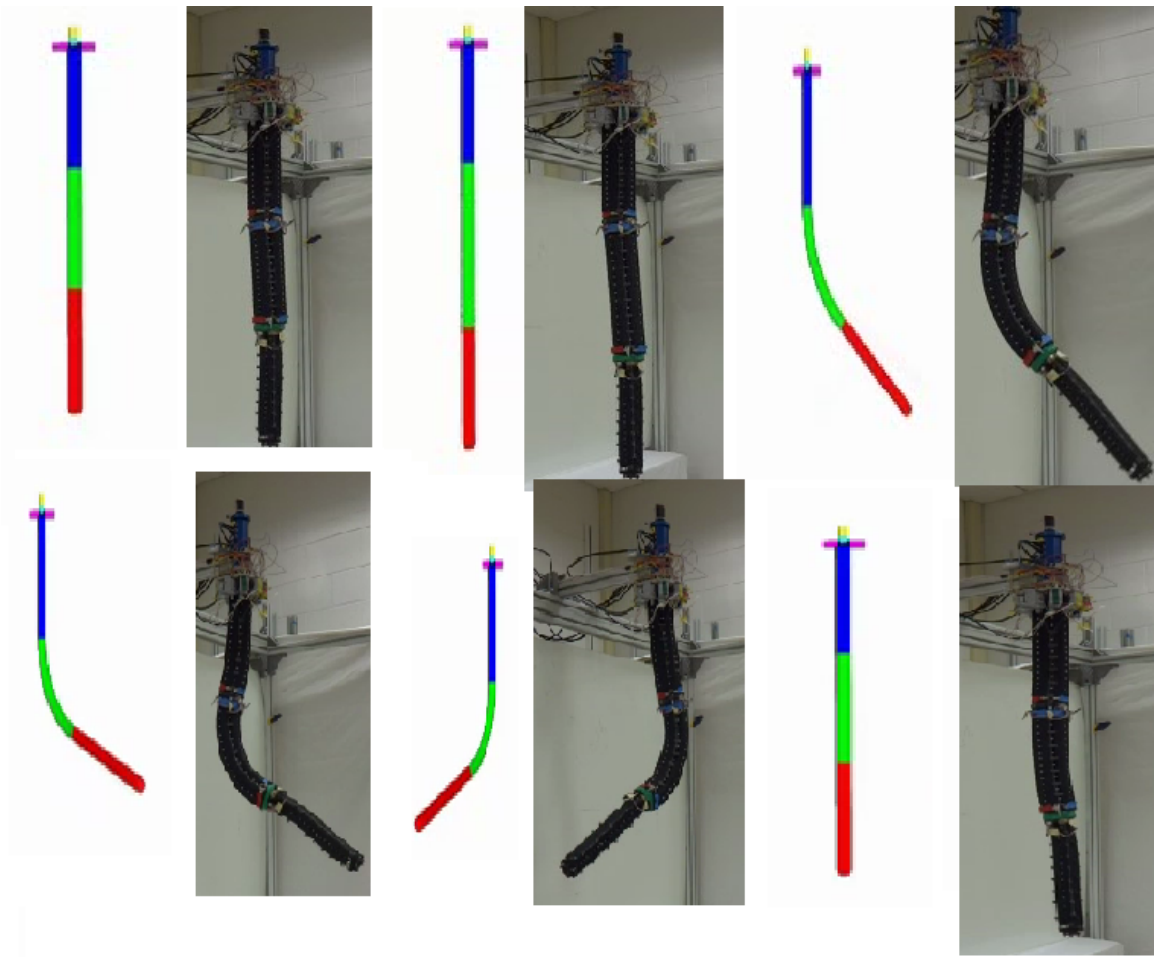


Figure 4.3: OctArm Validation of Midsection [33]

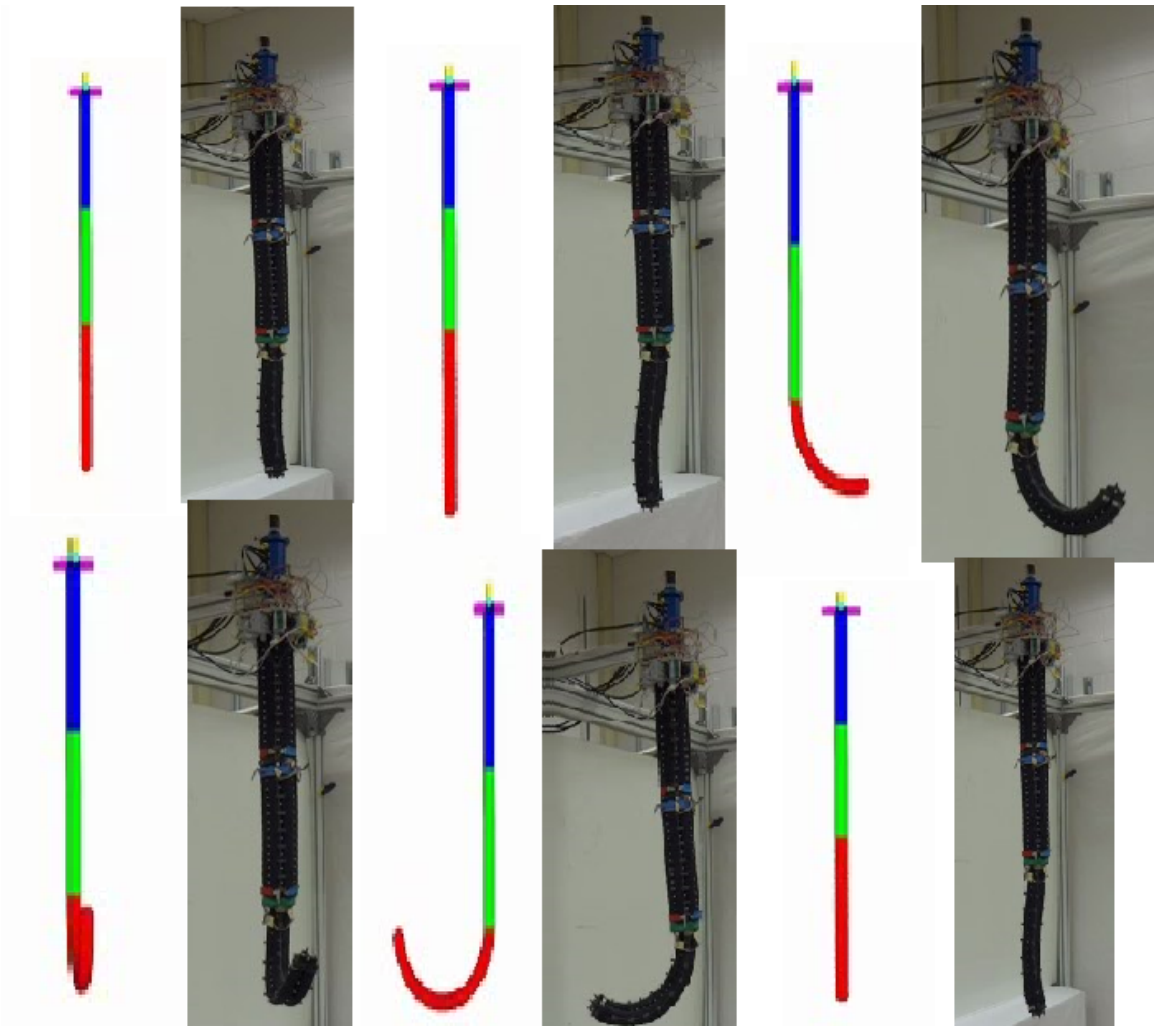


Figure 4.4: OctArm Validation of Tip Section [34]

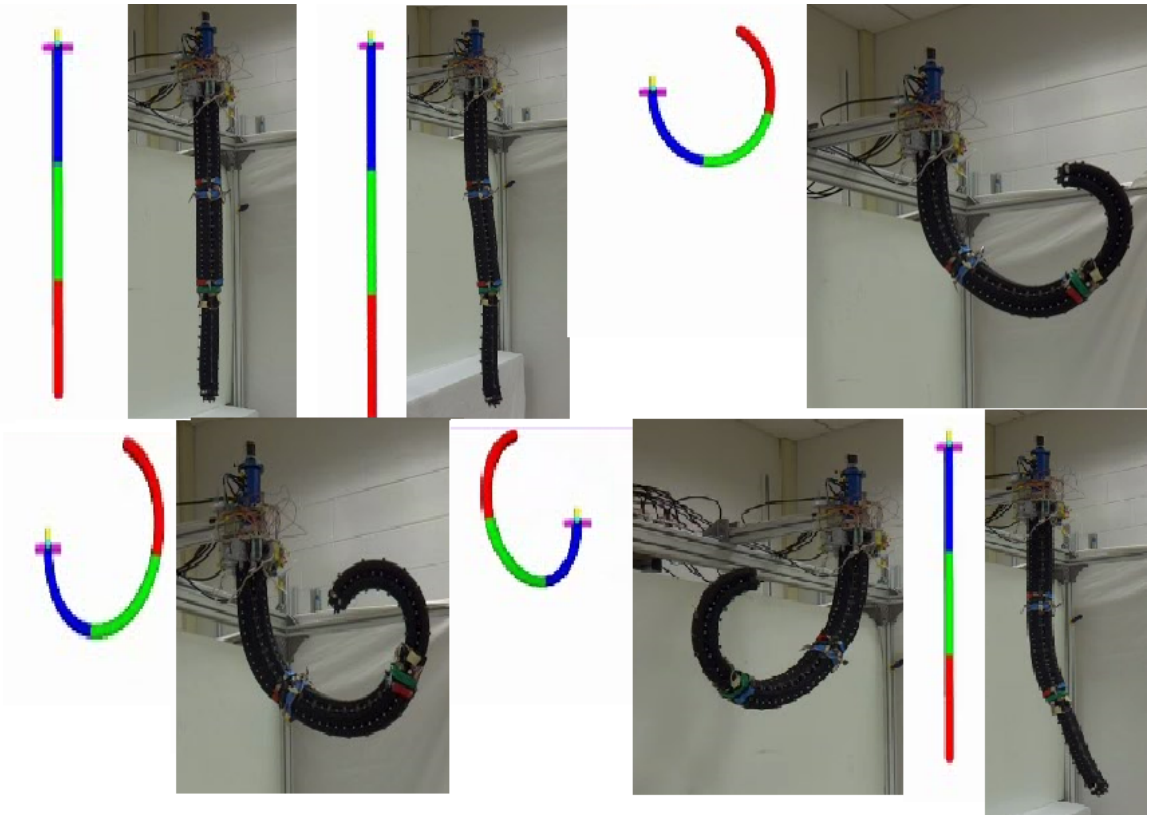


Figure 4.5: OctArm Validation [31]

4.3 Tendril Hardware

The tendril is a thin tendon actuated continuum robot. Its backbone is made up of three carbon fiber concentric tubes, with the distal and midsection tubes contracting into each other and the base section tube, respectively. The tubes are spring loaded. Like the OctArm, the tendril uses three actuators set 120˚ apart. In this case, the actuators are servo motors that pull the tendons, so the tendril cannot extend it can only compress. To conveniently route the tendons, each set of tendons is offset 40˚ from the previous section. The base section is controlled by motor 0, 1 and 2 (at 0˚, 120˚, 240˚), the midsection is controlled by motor 3, 4 and 5 (at 40˚, 160˚, 280˚), and the distal section is controlled by motor 6, 7 and 8 (at 80˚, 200˚, 320˚) as portrayed in Figure 4.6 [41]. The actuator package is shown in Figure 4.7. The base section is constructed of 3D printed spacers glued to carbon fiber tubing and can curve, but does not have the capability to compress. The remaining two sections are constructed with springs between the spacers which allows for compressibility, as noted above. The servo motors are connected to an Arduino Mega connected to a PC via serial connection. Unlike the OctArm, which had (via string encoders) feedback of shape, the tendril's only feedback was via tendon tension sensing (on each of the nine tendons).

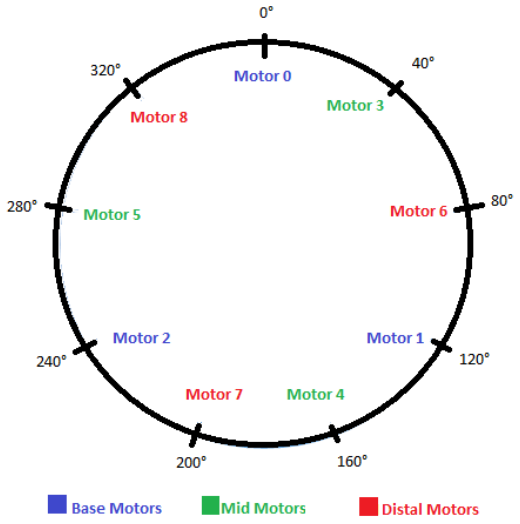


Figure 4.6: Tendril Motor Layout

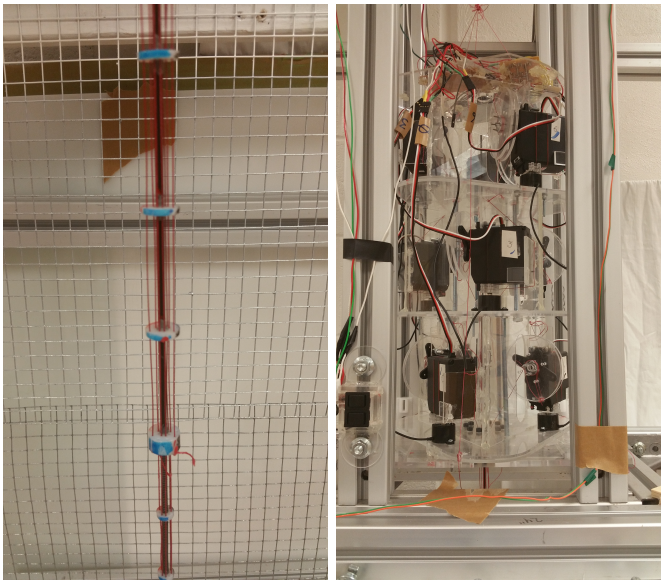


Figure 4.7: Tendril Robot

4.4 Tendril Validation

Unlike the OctArm, the tendril control system was not based on a kinematic model using s, κ, ϕ as inputs. So a new primitive mapping from s, κ, ϕ had to be created to move the tendril. In this work, only κ and ϕ were used to drive the tendril, where κ determined the tension a given set of motors would regulate to and ϕ determined which subset of motors needed to run. Each section's curvature is converted into a percentage of maximum tension by dividing by the maximum curvature for each section of the tendril. The maximum curvature was estimated by exerting maximum tension on the tendril and comparing it to the X3D model. Once they matched the κ value from the feedback section of the user interface was set as the maximum. Then to find the tension to be applied, the percentage of curvature is multiplied by the maximum tension value. This tension value is then split between the motors as a function of distance between them, i.e. if ϕ is halfway between motor 0 and motor 1, half of the tension is applied to motor 0 and half to motor 1.

A test of this primitive mapping was able to show that the tendril, under user command of the interface developed in this thesis responded as expected to the interface and was able to intuitively move all three sections. The S-Bend shape in Figure 4.8 demonstrates this basic capability. However, the tendril movements in some cases lagged behind the commanded curvature due in part from low resolution, a lack of calibration in the primitive mapping and the inability of the existing tendril

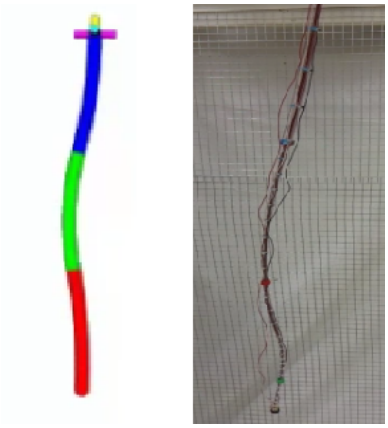


Figure 4.8: Tendril S Bend [35]

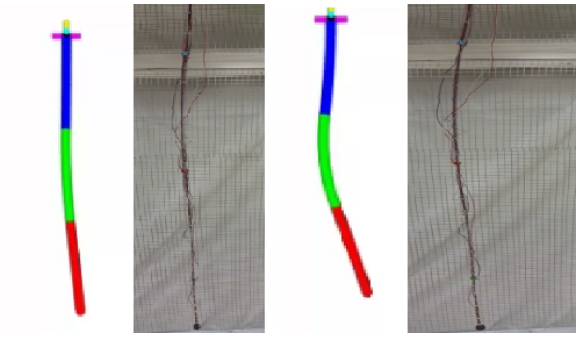


Figure 4.9: Limits on achievable curvatures (due to tendril hardware) [30]

servo actuators to move the base section, in particular, beyond a low curvature. The tendril matched the first curve in Figure 4.9 well, but it did not respond past that point.

Chapter 5

Conclusion

This thesis presents a novel and practical teleoperation interface for continuum robots. The underlying idea is for the human operator to directly manipulate a truly continuum, i.e. kinematically similar, model of a continuum robot. Previous approaches have constrained the operator to manipulate nonintuitive, rigid-linked kinematically dissimilar input devices. The interface herein was designed using a server client model. The server renders an X3D continuum 3D model, moves the 3D model according to the kinematic model and sends the configuration parameters to the client (the physical continuum robot).

5.1 Summary of Contributions

Very little work has been done in the field of teleoperation of continuum robots; this work introduces a new concept for continuum teleoperation. Previous approaches fail to provide a natural mapping. Each input device has fewer degrees-of-freedom than the continuum robot being controlled. This degree-of-freedom mismatch is seen as one of the greatest obstacles to successful teleoperation of continuum robots.

For the first time a continuum surrogate software interface was developed to create a natural degree-of-freedom mapping to a continuum robot. Previous controllers were developed, but the operators had kinematically dissimilar interfaces which required extensive practice to gain enough familiarity to operate effectively. The graphical user interface presented in this thesis provides an intuitive natural mapping that is synthesized and implemented using the server client model. This

approach is easier to use which provides for a better user experience overall.

Communication with the robot is simplified, because complex conversions are not required prior to issuing commands. The server software loads the graphical user interface, runs necessary code to render the graphics model of the robot and sends the appropriate data to the client. After changes are made in the server software run user interface, the client then reads a message from the server, interpreting the message and then directly writing the corresponding values to the hardware in order to physically move the robot. The user interface provides a low weight easy to access platform.

This interface was developed using the new declarative HTML5 standard for 3D graphics, X3D, and was validated with two different continuum robotic systems. Both the tendril and OctArm systems were manipulated using this software. These systems were originally designed to be operated by two different software suites, however this software was found to be capable of interfacing with both, allowing for easier user control.

5.2 Future Work

This teleoperative interface uses open-loop control, there is no feedback to the model. Feedback could be added to improve both the accuracy of the control and the graphical model. In the case of the OctArm, the C dll file could incorporate `hil_analog_read()` functions and then send these back over the socket-io interface to the server to update the current state. The `TendrilClient` class is already set up to read information back from the serial port using the `PortReader` class. To add feedback, the Arduino code would need to be updated and the information from `PortReader` would need to be sent back using socket-io to the server.

The `TendrilSystem` class makes numerous assumptions about how the tendril behaves kinematically due to a lack of a working kinematic model for the mechanical system of the tendril, which at the time of writing is under development. As a consequence, the s parameter is also completely ignored in the currently implemented model, so this needs to be added as well.

When this project began in October 2014, there was very little information on the web about X3D. In the several weeks prior to defense of this thesis, much more has been published in terms of API documentation, examples and tutorials as well as additional X3D features. Chief among these new features is a click to select feature on specific X3D object. Implementing this would remove

the need for checkboxes allowing for a cleaner UI and make a touch screen interface practical. Since this interface is written using JavaScript, CSS3, HTML5 and Java this would extend easily to an Android environment allowing for a mobile app to be developed quickly. Following these updates, experiments will be developed where users will perform tasks with this interface and different input devices so as to compare this approach to previous teleoperation approaches to evaluate the user experience.

Appendices

Appendix A

Glossary of Programming Terms

Application Program Interface (API) An API is an interface that contains a set of functions, protocols and tools for two different software programs to communicate with each other.

Attribute Part of an HTML or XML element, that describes some characteristic of that element. They are put inside the start tag and defined using name value pairs. For example, in <`transform id="b0" ></transform`> the element is transform and the attribute is id (name: `id`, value: `"b0"`).

Class A class is a file that is a template for an object. A class consists of fields, methods, other classes, or any combination of the three.

Constructor A method in a class that creates an object. A constructor always has the same name as the class.

Dependency Anything a program needs to either execute a compile. Usually these are libraries or APIs.

Deprecated Code that the developer no longer supports and is labeled use at your own risk. It is not removed to allow for some backwards compatibility but is not guaranteed to work.

Dynamic Link Library (DLL) A library that can be loaded at runtime when an application needs it. This saves memory usage and allows for multiple programs to use the same library file concurrently.

Encapsulation The ability to hide fields and implementation in a class, so the internals can change without having to worry about changing code outside the class.

Field A variable associated with a class.

Integrated Development Environment (IDE) A software application used for software development that consists of an editor, compiler, build automation tools and a debugger.

Instance Another term for an object. An instance variable is defined in a class and each object has a separate and unique copy associated with it.

Instantiate The process of creating an instance or object.

Key-Value Pair A set of associated data items. The key is used as an identifier and the value is the data that is being identified.

Method A function associated with a class.

Object An object is a realization of a class. In the snippet `Quanser Q8 = new Quanser()`, `Q8` is an object, `Quanser` is a class, and `Quanser()` is a constructor.

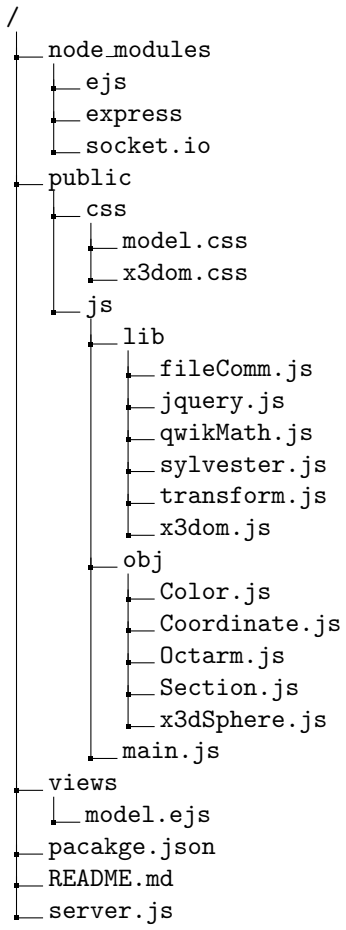
Sandbox A runtime environment that is isolated as a security mechanism to prevent any risk of harm to the host machine.

Stringify A built in JSON method that converts a JavaScript Object to a JSON string.

Tag Tags are sets of </>in HTML or XML. They are used to define elements.

Appendix B

Server Code File Listing



The node_modules folder contains the three node modules: ejs, express and socket-io. The

public folder contains all of the supporting files needed for the displayed webpage and is split into two folders css and js. The css folder contains the two CSS stylesheets: `model.css` and `x3dom.css`. The js folder contains the bulk of the JavaScript code. The JavaScript libraries described in Section 2.4 are in the lib folder and the JavaScript objects described in Section 2.2. Lastly there is the entry point JavaScript file `main.js`. The views folder contains the `model.ejs` file that defines the UI. There are three "loose" files: `package.json`, `README.md`, and `server.js`. The `package.json` file is used by the node js runtime environment to list the dependent modules housed in the node_modules folder. Lastly `server.js` is the file that runs the server necessary to start the program. To start command prompt must be opened to the directory containing `server.js` and then the command `node server.js` command must be executed. **Note:** node js must be installed, it can be found at [8].

Appendix A model.css

```
h2 {
  text-align:center;
}

div {
  margin:auto;
  text-align:center;
}

#x3dcontent {
  margin:auto;
  height:37.0370%; /*Full HD 800px/1080px = 0.74074em = 74.074%*/
  width: 37.0370%; /* 26.0415% Full HD 1000px/1920px = 0.52083em = 52.083%*/
  background-color: #FFFFFF;
  border: 10px solid #522D80;
  border-radius: 5px;
}

.row{
  display: table;
  width: 55%;
  border: 10px solid #522D80;
  border-radius: 5px;
}

.column{
  display: table-cell;
  background-color: white;
  border: 2.5px solid #522D80;
}

html {
  background-color: #F66733;
}

label {
```

```
37    font-weight: bold;
38  }
39
40  p {
41    font-family: Helvetica;
42  }
```

Appendix B fileComm.js

```
1  //fileComm.js
2
3  var fileComm = {
4    download : function download(strData, strFileName, strMimeType) {
5      var D= document,
6        A= arguments,
7        a = D.createElement("a"),
8        d = A[0],
9        n = A[1],
10       t = A[2] || "text/plain";
11
12       a.href = "data: "+ strMimeType + "charset=utf-8, " + escape(strData);
13
14       if(window.MSBlobBuilder){
15         var bb = new MSBlobBuilder();
16         bb.append(strData);
17         return navigator.msSaveBlob(bb, strFileName);
18       }
19
20       if('download' in a){
21         a.setAttribute("download",n);
22         a.innerHTML = "downloading...";
23         D.body.appendChild(a);
24         setTimeout(function(){
25           var e = D.createEvent("MouseEvents");
26           e.initMouseEvent("click", true, false, window, 0, 0 , 0, 0, 0, false, false, false,
                 false, 0, null);
27           a.dispatchEvent(e);
28           D.body.removeChild(a);
29
30         },66);
31         return true;
32       }
33   },
34
35   openSocket : function openSocket(){
```

```
36      socket = new WebSocket('ws://localhost:8000/echo');
37      socket.onopen = function(){
38        alert("socket opened");
39      };
40      socket.onmessage = function(str){
41        alert('Got reply '+str);
42      };
43    },
44
45    formatData : function formatData(robot) {
46      var obj = {
47        "params":
48        [
49          {"base_s" : robot.base.length},
50          {"base_k" : robot.base.k},
51          {"base_phi" : robot.base.phi},
52          {"mid_s" : robot.mid.length},
53          {"mid_k" : robot.mid.k},
54          {"mid_phi" : robot.mid.phi},
55          {"tip_s" : robot.tip.length},
56          {"tip_k" : robot.tip.k},
57          {"tip_phi" : robot.tip.phi}
58        ]
59      };
60      return JSON.stringify(obj);
61    },
62
63    sendData : function sendData(data) {
64      var socket = io ();
65      socket.emit('message', data);
66    }
67  }
```

Appendix C qwikMath.js

```
1  //qwikMath.js
2
3  var qwikMath = {
4
5    dist: function dist(obj1, obj2){
6      var x1 = obj1.getX();
7      var y1 = obj1.getY();
8      var z1 = obj1.getZ();
9      var x2 = obj2.getX();
10     var y2 = obj2.getY();
11     var z2 = obj2.getZ();
12     var ssx = (x2-x1)*(x2-x1);
13     var ssy = (y2-y1)*(y2-y1);
14     var ssz = (z2-z1)*(z2-z1);
15
16     return Math.sqrt(ssx+ssy+ssz);
17   },
18
19   formatAngles: function formatAngles(angle) {
20     if(angle >= 360){
21       return Math.round(angle) - 360;
22     } else if(angle <= -360){
23       return Math.round(angle) + 360;
24     } else{
25       return Math.round(angle);
26     }
27   }
28
29 }
```

Appendix D transform.js

```
1  //transform.js - based on Denavit-Hartenberg Algorithm
2
3  var transform = {
4   createMatrix : function createMatrix(s, k, phi){
5
6     var H3d = $M([
7       [Math.cos(phi), -Math.sin(phi)*Math.cos(s*k), Math.sin(phi)*Math.sin(s*k),
            (1/k)*(Math.sin(phi)*(1-Math.cos(s*k)))],
8       [Math.sin(phi), Math.cos(phi)*Math.cos(s*k), -Math.cos(phi)*Math.sin(s*k),
            -(1/k)*(Math.cos(phi)*(1-Math.cos(s*k)))],
9       [           0,              Math.sin(s*k),             Math.cos(s*k),
            (1/k)*Math.sin(s*k)],
10      [           0,                          0,                         0,
                                                 1]
11    ]);
12
13    return H3d;
14  },
15
16
17  kinematics : function kinematics(base, mid, tip){
18
19    for (var i = 0; i < base.objs.length; i++) {
20      var Hb = transform.createMatrix(base.d*(i+1), base.k, base.phi);
21      base.objs[i].setX(Hb.e(1,4));
22      base.objs[i].setY(Hb.e(2,4));
23      base.objs[i].setZ(Hb.e(3,4));
24    }
25
26
27    for (var i = 0; i < mid.objs.length; i++) {
28      var Hm = transform.createMatrix(mid.d*(i+1), mid.k, mid.phi);
29      var m = Hb.multiply(Hm);
30      var mres = m.minor(1,4,3,1);
31      mid.objs[i].setX(mres.e(1,1));
32      mid.objs[i].setY(mres.e(2,1));
```

```
      mid.objs[i].setZ(mres.e(3,1));

    }

    for (var i = 0; i < base.objs.length; i++) {
      var Ht = transform.createMatrix(tip.d*(i+1), tip.k, tip.phi);
      var t = Hb.multiply(Hm).multiply(Ht);
      var tres = t.minor(1,4,3,1);
      tip.objs[i].setX(tres.e(1,1));
      tip.objs[i].setY(tres.e(2,1));
      tip.objs[i].setZ(tres.e(3,1));

    }

  },


}
```

Appendix E Color.js

```
1  //Color.js
2
3  function Color(string){
4    var pos = string.split(" ");
5    var counter = 0, flag = false;
6    for (var i = 0; i < pos.length; i++) {
7      if( isNumeric(pos[i]) ){
8        counter++;
9        switch(counter){
10         case 1: this.r = +pos[i]; break;
11         case 2: this.g = +pos[i]; break;
12         case 3: this.b = +pos[i]; flag = true; break;
13       }
14       if(flag){
15         break;
16       }
17     }
18   }
19
20   this.getR = function getR(){
21     return this.r;
22   }
23
24   this.setR = function setR(r){
25     this.r = r;
26   }
27
28   this.getG = function getG(){
29     return this.g;
30   }
31
32   this.setG = function setG(g){
33     this.g = g;
34   }
35
36   this.getB = function getB(){
```

```
37      return this.b;
38    }
39
40    this.setB = function setB(b){
41      this.b = b;
42    }
43
44  }
```

Appendix F Coordinate.js

```
1  //Coordinates.js
2
3  function Coordinate(string){
4    var pos = string.split(" ");
5    var counter = 0, flag = false;
6    for (var i = 0; i < pos.length; i++) {
7      if( isNumeric(pos[i]) ){
8        counter++;
9        switch(counter){
10         case 1: this.x = +pos[i]; break;
11         case 2: this.y = +pos[i]; break;
12         case 3: this.z = +pos[i]; flag = true; break;
13       }
14       if(flag){
15         break;
16       }
17     }
18   }
19
20   this.getX = function getX(){
21     return this.x;
22   }
23
24   this.setX = function setX(x){
25     this.x = x;
26   }
27
28   this.getY = function getY(){
29     return this.y;
30   }
31
32   this.setY = function setY(y){
33     this.y = y;
34   }
35
36   this.getZ = function getZ(){
```

```
37      return this.z;
38    }
39
40    this.setZ = function setZ(z){
41      this.z = z;
42    }
43
44
45  }
46
47  function isNumeric(string){
48    if(strcmp(string, ""))
49      return 0;
50    return 1;
51  }
52
53  function strcmp(str1, str2){
54    if(!str1.localeCompare(str2)) //localeCompare returns 0 if exact match
55      return true;
56    return false;
57  }
```

Appendix G Octarm.js

```
//Octarm.js
function Octarm() {
  this.anchor = new x3dSphere('a');
  this.base = new Section("b", this.anchor);
  this.mid = new Section("m", this.base.objs[this.base.n-1]);
  this.tip = new Section("t", this.mid.objs[this.mid.n-1]);
  this.n = this.base.n;

  this.curve = function curve(section, sign){

    switch(section){

      case 'b':
        this.base.k+=sign*0.01176;
        transform.kinematics(this.base, this.mid, this.tip);
        break;

      case 'm':
        this.mid.k+=sign*0.01176;
        transform.kinematics(this.base, this.mid, this.tip);

        break;

      case 'bm':
        this.base.k+=sign*0.01176;
        this.mid.k+=sign*0.001176;
        transform.kinematics(this.base, this.mid, this.tip);

        break;

      case 'mt':
        this.mid.k+=sign*0.01176;
        this.tip.k+=sign*0.01176;
        transform.kinematics(this.base, this.mid, this.tip);
        break;

```

```
37        case 't':
38          this.tip.k+=sign*0.01176;
39          transform.kinematics(this.base, this.mid, this.tip);
40          break;
41
42        case 'bmt':
43          this.base.k+=sign*0.01176;
44          this.mid.k+=sign*0.01176;
45          this.tip.k+=sign*0.01176;
46          transform.kinematics(this.base, this.mid, this.tip);
47          break;
48        default:
49          alert("Please select a section");
50
51      }
52
53    }
54
55    this.extend = function(section, sign){
56
57      switch(section){
58
59        case 'b':
60          this.base.d+=sign*0.00166;
61          transform.kinematics(this.base, this.mid, this.tip);
62          break;
63
64        case 'm':
65          this.mid.d+=sign*0.00166;
66          transform.kinematics(this.base, this.mid, this.tip);
67
68          break;
69
70        case 'bm':
71          this.base.d+=sign*0.00166;
72          this.mid.d+=sign*0.00166;
73          transform.kinematics(this.base, this.mid, this.tip);
74
75          break;
```

```
76
77        case 'mt':
78          this.mid.d+=sign*0.00166;
79          this.tip.d+=sign*0.00166;
80          transform.kinematics(this.base, this.mid, this.tip);
81          break;
82
83        case 't':
84          this.tip.d+=sign*0.00166;
85          transform.kinematics(this.base, this.mid, this.tip);
86          break;
87
88        case 'bmt':
89          this.base.d+=sign*0.00166;
90          this.mid.d+=sign*0.00166;
91          this.tip.d+=sign*0.00166;
92          transform.kinematics(this.base, this.mid, this.tip);
93          break;
94        default:
95          alert("Please select a section");
96      }
97    }
98
99    this.orient = function(section, sign){
100
101     switch(section){
102
103       case 'b':
104         this.base.phi+=sign*0.0872;
105         transform.kinematics(this.base, this.mid, this.tip);
106         break;
107
108       case 'm':
109         this.mid.phi+=sign*0.0872;
110         transform.kinematics(this.base, this.mid, this.tip);
111
112         break;
113
114       case 'bm':
```

```
115         this.base.phi+=sign*0.0872;
116         this.mid.phi+=sign*0.0872;
117         transform.kinematics(this.base, this.mid, this.tip);
118
119         break;
120
121       case 'mt':
122         this.mid.phi+=sign*0.0872;
123         this.tip.phi+=sign*0.0872;
124         transform.kinematics(this.base, this.mid, this.tip);
125         break;
126
127       case 't':
128         this.tip.phi+=sign*0.0872;
129         transform.kinematics(this.base, this.mid, this.tip);
130         break;
131
132       case 'bmt':
133         this.base.phi+=sign*0.0872;
134         this.mid.phi+=sign*0.0872;
135         this.tip.phi+=sign*0.0872;
136         transform.kinematics(this.base, this.mid, this.tip);
137         break;
138       default:
139         alert("Please select a section");
140     }
141   }
142 }
```

Appendix H Section.js

```
1  //Section.js
2
3  function Section(type, anchorObj) {
4    this.n =(document.querySelectorAll('transform').length -6) /3;
5    this.type = type;
6    this.objs = [];
7    this.k = 0.0012;
8    this.phi = 0;
9    for (var i = 0; i < this.n; i++) {
10     this.objs[i] = new x3dSphere(type+i);
11   }
12
13   this.anchor = anchorObj;
14
15   this.startpt = this.objs[0];
16
17   this.radius = this.objs[0].radius;
18
19   this.getRadius = function getRadius() {
20     this.radius;
21   }
22
23   this.setRadius = function setRadius(rad){
24     for (var i = 0; i < this.objs.length; i++) {
25       this.objs[i].radius = rad;
26     }
27     this.radius = rad;
28   }
29
30   this.color = this.objs[0].color;
31   this.length = qwikMath.dist(this.objs[this.n-1], this.anchor.coords);
32   this.d = this.length/this.n;
33
34 }
```

Appendix I x3dSphere.js

```
1  //x3dSphere.js
2
3  function x3dSphere(id) {
4    this.id = "#"+id;
5    this.position = $(this.id).attr('translation');
6    this.coords = new Coordinate(this.position);
7    this.radius = Number($(this.id).children().children("sphere").attr('radius'));
8    this.color = new
         Color($(this.id).children().children("appearance").children().attr('diffuseColor'));
9
10   this.getPosition = function getPosition(){
11     return this.position;
12   }
13
14   this.setPosition = function setPosition(x,y,z){
15     this.coords.setX(x);
16     this.coords.setY(y);
17     this.coords.setZ(z);
18     this.position = this.coords.x + " " + this.coords.y + " " + this.coords.z;
19     $(this.id).attr('translation', this.position);
20   }
21
22   this.getX = function getX(){
23     return this.coords.x;
24   }
25
26   this.setX = function setX(x){
27     this.coords.setX(x);
28     this.position = this.coords.x + " " + this.coords.y + " " + this.coords.z;
29     $(this.id).attr('translation', this.position);
30   }
31
32   this.addX = function addX(x){
33     this.coords.setX(this.coords.x+x);
34     this.position = this.coords.x + " " + this.coords.y + " " + this.coords.z;
35     $(this.id).attr('translation', this.position);
```

```
36    }
37
38    this.getY = function getY(){
39      return this.coords.y;
40    }
41
42    this.setY = function setY(y){
43      this.coords.setY(y);
44      this.position = this.coords.x + " " + this.coords.y + " " + this.coords.z;
45      $(this.id).attr('translation', this.position);
46    }
47
48    this.addY = function addY(y){
49      this.coords.setY(this.coords.y+y);
50      this.position = this.coords.x + " " + this.coords.y + " " + this.coords.z;
51      $(this.id).attr('translation', this.position);
52    }
53
54    this.getZ = function getZ(){
55      return this.coords.z;
56    }
57
58    this.setZ = function setZ(z){
59      this.coords.setZ(z);
60      this.position = this.coords.x + " " + this.coords.y + " " + this.coords.z;
61      $(this.id).attr('translation', this.position);
62    }
63
64    this.addZ = function addZ(z){
65      this.coords.setZ(this.coords.z+z);
66      this.position = this.coords.x + " " + this.coords.y + " " + this.coords.z;
67      $(this.id).attr('translation', this.position);
68    }
69
70  }
```

Appendix J main.js

```
1  //main.js
2  var timeout;
3  $(document).ready(function(){
4
5    var robot = new Octarm();
6
7    transform.kinematics(robot.base, robot.mid, robot.tip);
8    $("#blength").text("Length: " + robot.base.length +"m");
9    $("#bcurve").text("Curvature: " + robot.base.k);
10   $("#borient").text("Orientation: " + robot.base.phi*180/Math.PI +" degrees");
11   $("#mlength").text("Length: " + robot.mid.length +"m");
12   $("#mcurve").text("Curvature: " + robot.mid.k);
13   $("#morient").text("Orientation: " + robot.mid.phi*180/Math.PI +" degrees");
14   $("#tlength").text("Length: " + robot.tip.length.toFixed(3) +"m");
15   $("#tcurve").text("Curvature: " + robot.tip.k);
16   $("#torient").text("Orientation: " + robot.tip.phi*180/Math.PI +" degrees");
17
18   $("#orientCCW").click(function() {
19     var sec = '';
20
21     if($("#orientBase").prop("checked")){
22       sec+='b';
23     }
24
25     if($("#orientMid").prop("checked")){
26       sec+='m';
27     }
28
29     if($("#orientTip").prop("checked")){
30       sec+='t';
31     }
32     //alert(sec);
33     robot.orient(sec, -1);
34
35     $("#borient").text("Orientation: " + qwikMath.formatAngles(robot.base.phi*180/Math.PI) +"
         degrees");
```

```
36      $("#morient").text("Orientation: " + qwikMath.formatAngles(robot.mid.phi*180/Math.PI) +"
            degrees");
37      $("#torient").text("Orientation: " + qwikMath.formatAngles(robot.tip.phi*180/Math.PI) +"
            degrees");
38      fileComm.sendData(fileComm.formatData(robot));
39    });
40
41    $("#orientCW").click(function() {
42      var sec = '';
43
44      if($("#orientBase").prop("checked")){
45        sec+='b';
46      }
47
48      if($("#orientMid").prop("checked")){
49        sec+='m';
50      }
51
52      if($("#orientTip").prop("checked")){
53        sec+='t';
54      }
55      //alert(sec);
56      robot.orient(sec, 1);
57
58      $("#borient").text("Orientation: " + qwikMath.formatAngles(robot.base.phi*180/Math.PI) +"
            degrees");
59      $("#morient").text("Orientation: " + qwikMath.formatAngles(robot.mid.phi*180/Math.PI) +"
            degrees");
60      $("#torient").text("Orientation: " + qwikMath.formatAngles(robot.tip.phi*180/Math.PI) +"
            degrees");
61      fileComm.sendData(fileComm.formatData(robot));
62    });
63
64    $("#extend").click(function() {
65      var sec = '';
66
67      if($("#extendBase").prop("checked")){
68        sec+='b';
69      }
```

```
70
71      if($("#extendMid").prop("checked")){
72        sec+='m';
73      }
74
75      if($("#extendTip").prop("checked")){
76        sec+='t';
77      }
78      //alert(sec);
79      robot.extend(sec, 1);
80      robot.base.length = robot.base.d*robot.base.n;
81      robot.mid.length = robot.mid.d*robot.mid.n;
82      robot.tip.length = robot.tip.d*robot.tip.n;
83
84      $("#blength").text("Length: " + robot.base.length.toFixed(3) +"m");
85      $("#mlength").text("Length: " + robot.mid.length.toFixed(3) +"m");
86      $("#tlength").text("Length: " + robot.tip.length.toFixed(3) +"m");
87      fileComm.sendData(fileComm.formatData(robot));
88    });
89
90    $("#compress").click(function() {
91      var sec = '';
92
93      if($("#extendBase").prop("checked")){
94        sec+='b';
95      }
96
97      if($("#extendMid").prop("checked")){
98        sec+='m';
99      }
100
101     if($("#extendTip").prop("checked")){
102       sec+='t';
103     }
104     //alert(sec);
105     robot.extend(sec, -1);
106     robot.base.length = robot.base.d*robot.base.n;
107     robot.mid.length = robot.mid.d*robot.mid.n;
108     robot.tip.length = robot.tip.d*robot.tip.n;
```

```
109
110     $("#blength").text("Length: " + robot.base.length.toFixed(3) +"m");
111     $("#mlength").text("Length: " + robot.mid.length.toFixed(3) +"m");
112     $("#tlength").text("Length: " + robot.tip.length.toFixed(3) +"m");
113     fileComm.sendData(fileComm.formatData(robot));
114   });
115
116   $("#curveL").click(function() {
117
118     var sec = '';
119     if($("#curveBase").prop("checked")){
120       sec += 'b';
121     }
122
123     if($("#curveMid").prop("checked")){
124       sec += 'm';
125     }
126
127     if($("#curveTip").prop("checked")){
128       sec += 't';
129     }
130
131     robot.curve(sec, -1);
132     $("#bcurve").text("Curvature: " + +robot.base.k.toFixed(4));
133     $("#mcurve").text("Curvature: " + +robot.mid.k.toFixed(4));
134     $("#tcurve").text("Curvature: " + +robot.tip.k.toFixed(4));
135     fileComm.sendData(fileComm.formatData(robot));
136   });
137
138   $("#curveR").click(function() {
139
140     var sec = '';
141     if($("#curveBase").prop("checked")){
142       sec += 'b';
143     }
144
145     if($("#curveMid").prop("checked")){
146       sec += 'm';
147     }
```

```
148
149     if($("#curveTip").prop("checked")){
150       sec += 't';
151     }
152
153     robot.curve(sec, 1);
154     $("#bcurve").text("Curvature: " + +robot.base.k.toFixed(4));
155     $("#mcurve").text("Curvature: " + +robot.mid.k.toFixed(4));
156     $("#tcurve").text("Curvature: " + +robot.tip.k.toFixed(4));
157     fileComm.sendData(fileComm.formatData(robot));
158   });
159
160   $("#reloadPage").click(function(){
161     history.go(0);
162   });
163
164   $("#writeFile").click(function(){
165     fileComm.fileWrite("TEST");
166   })
167
168   $("#downloadTxt").click(function(){
169     var string="Base: s="+robot.base.length+" k="+ robot.base.k+" | Mid: s="+robot.mid.length+"
          k="+robot.mid.k+" | Tip: s="+robot.tip.length+" k="+robot.tip.k;
170     fileComm.download(string, 'skdata.txt', 'text/plain');
171   });
172 });
```

Appendix K Model.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Ryan Scott X3D</title>
  <meta charset="UTF-8">
  <!--<link rel="stylesheet" type="text/css"
       href="http://www.x3dom.org/x3dom/release/x3dom.css"></link>-->
  <link rel="stylesheet" type="text/css" href="/css/model.css"></link>
  <!-- <link rel="stylesheet" href="/css/bootstrap.min.css" ></link> -->

  <script src="/js/lib/x3dom.js"></script>
  <script src="/js/lib/jquery.js"></script>
  <script src="/js/lib/qwikMath.js"></script>
  <script src="/js/lib/sylvester.js"></script>
  <script src="/js/lib/transform.js"></script>
  <script src="/js/lib/fileComm.js"></script>
  <script src="/js/obj/x3dSphere.js"></script>
  <script src="/js/obj/Coordinate.js"></script>
  <script src="/js/obj/Color.js"></script>
  <script src="/js/obj/Section.js"></script>
  <script src="/js/obj/Octarm.js"></script>
  <script src="/js/main.js"></script>
  <script src="https://cdn.socket.io/socket.io-1.2.0.js"></script>

</head>
<body>
  <h2> X3D Interface </h2>


  <div class="row">
    <div class="column">
      <h3> Base Properties </h3>
      <label for="bcolor"> Base Color </label>
      <input id="bcolor" type="color" value="#0000ff">
      <p id="blength">[ERROR]: No Base</p>
      <p id ="bcurve">[ERROR]: No Base</p>
```

```
36        <p id="borient">[ERROR]: No Base</p>
37        <input type="checkbox" id="extendBase">S</input>
38        <input type="checkbox" id="curveBase">&Kappa;</input>
39        <input type="checkbox" id="orientBase">&Phi;</input>
40      </div>
41
42      <div class="column">
43        <h3> Midsection Properties </h3>
44        <label for="mcolor"> Midsection Color </label>
45        <input id="mcolor" type="color" value="#00ff00">
46        <p id="mlength">[ERROR]: No Mid</p>
47        <p id="mcurve">[ERROR]: No Mid</p>
48        <p id="morient">[ERROR]:No Mid</p>
49        <input type="checkbox" id="extendMid">S</input>
50        <input type="checkbox" id="curveMid">&Kappa;</input>
51        <input type="checkbox" id="orientMid">&Phi;</input>
52      </div>
53
54      <div class="column">
55        <h3> Tip Properties </h3>
56        <label for="tcolor">Tip Color </label>
57        <input id="tcolor" type="color" value="#ff0000">
58        <p id=tlength>[ERROR]: No Tip</p>
59        <p id=tcurve>[ERROR]: No Tip</p>
60        <p id=torient>[ERROR]: No Tip</p>
61        <input type="checkbox" id="extendTip">S</input>
62        <input type="checkbox" id="curveTip">&Kappa;</input>
63        <input type="checkbox" id="orientTip">&Phi;</input>
64      </div>
65    </div>
66
67    <div class="ui">
68      <button id="extend"> Extend (S) &larr; &rarr; </button>
69      <button id="compress"> Compress (S) &rarr; &larr; </button>
70      <button id="curveL"> Curve (&Kappa;) &larr; </button>
71      <button id="curveR"> Curve (&Kappa;) &rarr; </button>
72      <button id="orientCW"> Orient (&Phi;) &#8635; </button>
73      <button id="orientCCW"> Orient (&Phi;) &#8634; </button>
74      <button id="reloadPage"> Home </button>
```

```
75      <button id="downloadTxt">Download Configuration</button>
76    </div>
77
78
79    <p id="outputbox"></p>
80    <div id="x3dcontent">
81    <x3d id="the_model" width="100%" height="100%">
82      <scene>
83        <!--Set Up Viewpoint along -y axis -->
84        <transform rotation = "1 0 0 1.57">
85          <transform rotation="1 0 1 3.14">
86            <Viewpoint position = "0 -0.5 1.5"></Viewpoint>
87          </transform>
88        </transform>
89        <!-- Draw Coordinate Axes -->
90        <!-- magenta -->
91        <transform id="xaxis" DEF="xaxis" translation="0 0 0">
92          <shape>
93            <appearance>
94              <material diffuseColor="1 0 1"></material>
95            </appearance>
96            <cylinder radius="0.01" height=".1"></cylinder>
97          </shape>
98        </transform>
99        <!-- cyan -->
100       <transform id="yaxis" DEF="yaxis" translation="0 0 0" rotation="0 0 1 1.57">
101         <shape>
102           <appearance>
103             <material diffuseColor="0 1 1"></material>
104           </appearance>
105           <cylinder radius="0.01" height=".1"></cylinder>
106         </shape>
107       </transform>
108       <!-- yellow -->
109       <transform id="zaxis" DEF="zaxis" translation="0 0 0" rotation="1 0 0 1.57">
110         <shape>
111           <appearance>
112             <material diffuseColor="1 1 0"></material>
113           </appearance>
```

```
114           <cylinder radius="0.01" height=".1"></cylinder>
115         </shape>
116       </transform>
117
118       <!-- Anchor Node -->
119       <transform id="a" DEF="a" translation='0 0 0'>
120         <shape>
121           <appearance>
122             <material diffuseColor='0 0 0'></material>
123           </appearance>
124
125           <sphere radius="0.0174"></sphere>
126         </shape>
127       </transform>
128       <!--Base Section Nodes-->
129
130       <transform id="b0" DEF="b0" translation='0 0 0.0070'>
131         <shape>
132           <appearance>
133             <material diffuseColor='0 0 1'></material>
134           </appearance>
135
136           <sphere radius="0.0174"></sphere>
137         </shape>
138       </transform>
139       <transform id="b1" DEF="b1" translation='0 0 0.0140'>
140         <shape>
141           <appearance>
142             <material diffuseColor='0 0 1'></material>
143           </appearance>
144
145           <sphere radius="0.0174"></sphere>
146         </shape>
147       </transform>
148
149       <transform id="b2" DEF="b2" translation='0 0 .0209'>
150         <shape>
151           <appearance>
152             <material diffuseColor='0 0 1'></material>
```

```
153           </appearance>
154
155           <sphere radius="0.0174"></sphere>
156         </shape>
157       </transform>
158
159       <transform id="b3" DEF="b3" translation='0 0 .0279'>
160         <shape>
161           <appearance>
162             <material diffuseColor='0 0 1'></material>
163           </appearance>
164
165           <sphere radius="0.0174"></sphere>
166         </shape>
167       </transform>
168       <transform id="b4" DEF="b4" translation='0 0 .0349'>
169         <shape>
170           <appearance>
171             <material diffuseColor='0 0 1'></material>
172           </appearance>
173
174           <sphere radius="0.0174"></sphere>
175         </shape>
176       </transform>
177
178       <transform id="b5" DEF="b5" translation='0 0 .0419'>
179         <shape>
180           <appearance>
181             <material diffuseColor='0 0 1'></material>
182           </appearance>
183
184           <sphere radius="0.0174"></sphere>
185         </shape>
186       </transform>
187
188       <transform id="b6" DEF="b6" translation='0 0 .0488'>
189         <shape>
190           <appearance>
191             <material diffuseColor='0 0 1'></material>
```

```
192           </appearance>
193
194           <sphere radius="0.0174"></sphere>
195         </shape>
196       </transform>
197       <transform id="b7" DEF="b7" translation='0 0 .0558'>
198         <shape>
199           <appearance>
200             <material diffuseColor='0 0 1'></material>
201           </appearance>
202
203           <sphere radius="0.0174"></sphere>
204         </shape>
205       </transform>
206
207       <transform id="b8" DEF="b8" translation='0 0 .0628'>
208         <shape>
209           <appearance>
210             <material diffuseColor='0 0 1'></material>
211           </appearance>
212
213           <sphere radius="0.0174"></sphere>
214         </shape>
215       </transform>
216
217       <transform id="b9" DEF="b0" translation='0 0 .0698'>
218         <shape>
219           <appearance>
220             <material diffuseColor='0 0 1'></material>
221           </appearance>
222
223           <sphere radius="0.0174"></sphere>
224         </shape>
225       </transform>
226
227       <transform id="b10" DEF="b2" translation='0 0 .0768'>
228         <shape>
229           <appearance>
230             <material diffuseColor='0 0 1'></material>
```

```
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="b11" DEF="b3" translation='0 0 .0837'>
    <shape>
      <appearance>
        <material diffuseColor='0 0 1'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>
  <transform id="b12" DEF="b4" translation='0 0 .0907'>
    <shape>
      <appearance>
        <material diffuseColor='0 0 1'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="b13" DEF="b5" translation='0 0 .0977'>
    <shape>
      <appearance>
        <material diffuseColor='0 0 1'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="b14" DEF="b6" translation='0 0 .1047'>
    <shape>
      <appearance>
        <material diffuseColor='0 0 1'></material>
```

```
270           </appearance>
271
272           <sphere radius="0.0174"></sphere>
273         </shape>
274       </transform>
275       <transform id="b15" DEF="b7" translation='0 0 .1116'>
276         <shape>
277           <appearance>
278             <material diffuseColor='0 0 1'></material>
279           </appearance>
280
281           <sphere radius="0.0174"></sphere>
282         </shape>
283       </transform>
284
285       <transform id="b16" DEF="b8" translation='0 0 .1186'>
286         <shape>
287           <appearance>
288             <material diffuseColor='0 0 1'></material>
289           </appearance>
290
291           <sphere radius="0.0174"></sphere>
292         </shape>
293       </transform>
294
295       <transform id="b17" DEF="b0" translation='0 0 .1256'>
296         <shape>
297           <appearance>
298             <material diffuseColor='0 0 1'></material>
299           </appearance>
300
301           <sphere radius="0.0174"></sphere>
302         </shape>
303       </transform>
304       <transform id="b18" DEF="b1" translation='0 0 .1326'>
305         <shape>
306           <appearance>
307             <material diffuseColor='0 0 1'></material>
308           </appearance>
```

```
309
310           <sphere radius="0.0174"></sphere>
311         </shape>
312       </transform>
313
314       <transform id="b19" DEF="b1" translation='0 0 .1396'>
315         <shape>
316           <appearance>
317             <material diffuseColor='0 0 1'></material>
318           </appearance>
319
320           <sphere radius="0.0174"></sphere>
321         </shape>
322       </transform>
323
324       <transform id="b20" DEF="b2" translation='0 0 .1465'>
325         <shape>
326           <appearance>
327             <material diffuseColor='0 0 1'></material>
328           </appearance>
329
330           <sphere radius="0.0174"></sphere>
331         </shape>
332       </transform>
333
334       <transform id="b21" DEF="b3" translation='0 0 .1535'>
335         <shape>
336           <appearance>
337             <material diffuseColor='0 0 1'></material>
338           </appearance>
339
340           <sphere radius="0.0174"></sphere>
341         </shape>
342       </transform>
343       <transform id="b22" DEF="b4" translation='0 0 .1605'>
344         <shape>
345           <appearance>
346             <material diffuseColor='0 0 1'></material>
347           </appearance>
```

```
348
349           <sphere radius="0.0174"></sphere>
350         </shape>
351       </transform>
352
353       <transform id="b23" DEF="b5" translation='0 0 .1675'>
354         <shape>
355           <appearance>
356             <material diffuseColor='0 0 1'></material>
357           </appearance>
358
359           <sphere radius="0.0174"></sphere>
360         </shape>
361       </transform>
362
363       <transform id="b24" DEF="b8" translation='0 0 .1744'>
364         <shape>
365           <appearance>
366             <material diffuseColor='0 0 1'></material>
367           </appearance>
368
369           <sphere radius="0.0174"></sphere>
370         </shape>
371       </transform>
372
373       <transform id="b25" DEF="b0" translation='0 0 .1814'>
374         <shape>
375           <appearance>
376             <material diffuseColor='0 0 1'></material>
377           </appearance>
378
379           <sphere radius="0.0174"></sphere>
380         </shape>
381       </transform>
382       <transform id="b26" DEF="b1" translation='0 0 .1884'>
383         <shape>
384           <appearance>
385             <material diffuseColor='0 0 1'></material>
386           </appearance>
```

```
387
388           <sphere radius="0.0174"></sphere>
389         </shape>
390       </transform>
391
392       <transform id="b27" DEF="b2" translation='0 0 .1954'>
393         <shape>
394           <appearance>
395             <material diffuseColor='0 0 1'></material>
396           </appearance>
397
398           <sphere radius="0.0174"></sphere>
399         </shape>
400       </transform>
401
402       <transform id="b28" DEF="b3" translation='0 0 .2024'>
403         <shape>
404           <appearance>
405             <material diffuseColor='0 0 1'></material>
406           </appearance>
407
408           <sphere radius="0.0174"></sphere>
409         </shape>
410       </transform>
411       <transform id="b29" DEF="b4" translation='0 0 .2093'>
412         <shape>
413           <appearance>
414             <material diffuseColor='0 0 1'></material>
415           </appearance>
416
417           <sphere radius="0.0174"></sphere>
418         </shape>
419       </transform>
420
421       <transform id="b30" DEF="b5" translation='0 0 .2163'>
422         <shape>
423           <appearance>
424             <material diffuseColor='0 0 1'></material>
425           </appearance>
```

```
426
427           <sphere radius="0.0174"></sphere>
428         </shape>
429       </transform>
430
431       <transform id="b31" DEF="b6" translation='0 0 .2233'>
432         <shape>
433           <appearance>
434             <material diffuseColor='0 0 1'></material>
435           </appearance>
436
437           <sphere radius="0.0174"></sphere>
438         </shape>
439       </transform>
440       <transform id="b32" DEF="b7" translation='0 0 .2303'>
441         <shape>
442           <appearance>
443             <material diffuseColor='0 0 1'></material>
444           </appearance>
445
446           <sphere radius="0.0174"></sphere>
447         </shape>
448       </transform>
449
450       <transform id="b33" DEF="b8" translation='0 0 .2372'>
451         <shape>
452           <appearance>
453             <material diffuseColor='0 0 1'></material>
454           </appearance>
455
456           <sphere radius="0.0174"></sphere>
457         </shape>
458       </transform>
459
460       <transform id="b34" DEF="b0" translation='0 0 .2442'>
461         <shape>
462           <appearance>
463             <material diffuseColor='0 0 1'></material>
464           </appearance>
```

```
465
466           <sphere radius="0.0174"></sphere>
467         </shape>
468       </transform>
469       <transform id="b35" DEF="b1" translation='0 0 .2512'>
470         <shape>
471           <appearance>
472             <material diffuseColor='0 0 1'></material>
473           </appearance>
474
475           <sphere radius="0.0174"></sphere>
476         </shape>
477       </transform>
478
479       <transform id="b36" DEF="b2" translation='0 0 .2582'>
480         <shape>
481           <appearance>
482             <material diffuseColor='0 0 1'></material>
483           </appearance>
484
485           <sphere radius="0.0174"></sphere>
486         </shape>
487       </transform>
488
489       <transform id="b37" DEF="b3" translation='0 0 .2652'>
490         <shape>
491           <appearance>
492             <material diffuseColor='0 0 1'></material>
493           </appearance>
494
495           <sphere radius="0.0174"></sphere>
496         </shape>
497       </transform>
498       <transform id="b38" DEF="b4" translation='0 0 .2721'>
499         <shape>
500           <appearance>
501             <material diffuseColor='0 0 1'></material>
502           </appearance>
503
```

```
      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="b39" DEF="b5" translation='0 0 .2791'>
    <shape>
      <appearance>
        <material diffuseColor='0 0 1'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="b40" DEF="b6" translation='0 0 .2861'>
    <shape>
      <appearance>
        <material diffuseColor='0 0 1'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>
  <transform id="b41" DEF="b7" translation='0 0 .2931'>
    <shape>
      <appearance>
        <material diffuseColor='0 0 1'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="b42" DEF="b8" translation='0 0 .3000'>
    <shape>
      <appearance>
        <material diffuseColor='0 0 1'></material>
      </appearance>

```

```
543           <sphere radius="0.0174"></sphere>
544         </shape>
545       </transform>
546
547       <transform id="b43" DEF="b1" translation='0 0 .3070'>
548         <shape>
549           <appearance>
550             <material diffuseColor='0 0 1'></material>
551           </appearance>
552
553           <sphere radius="0.0174"></sphere>
554         </shape>
555       </transform>
556
557       <transform id="b44" DEF="b1" translation='0 0 .3140'>
558         <shape>
559           <appearance>
560             <material diffuseColor='0 0 1'></material>
561           </appearance>
562
563           <sphere radius="0.0174"></sphere>
564         </shape>
565       </transform>
566
567
568       <!--Midsection Nodes-->
569
570       <transform id="m0" DEF="m0" translation='0 0 .3210'>
571         <shape>
572           <appearance>
573             <material diffuseColor='0 1 0'></material>
574           </appearance>
575
576           <sphere radius="0.0174"></sphere>
577         </shape>
578       </transform>
579       <transform id="m1" DEF="m1" translation='0 0 .3280'>
580         <shape>
581           <appearance>
```

```
582             <material diffuseColor='0 1 0'></material>
583           </appearance>
584
585           <sphere radius="0.0174"></sphere>
586         </shape>
587       </transform>
588
589       <transform id="m2" DEF="m2" translation='0 0 .3349'>
590         <shape>
591           <appearance>
592             <material diffuseColor='0 1 0'></material>
593           </appearance>
594
595           <sphere radius="0.0174"></sphere>
596         </shape>
597       </transform>
598
599       <transform id="m3" DEF="m3" translation='0 0 .3419'>
600         <shape>
601           <appearance>
602             <material diffuseColor='0 1 0'></material>
603           </appearance>
604
605           <sphere radius="0.0174"></sphere>
606         </shape>
607       </transform>
608       <transform id="m4" DEF="m4" translation='0 0 .3489'>
609         <shape>
610           <appearance>
611             <material diffuseColor='0 1 0'></material>
612           </appearance>
613
614           <sphere radius="0.0174"></sphere>
615         </shape>
616       </transform>
617
618       <transform id="m5" DEF="m5" translation='0 0 .3559'>
619         <shape>
620           <appearance>
```

```
621             <material diffuseColor='0 1 0'></material>
622           </appearance>
623
624           <sphere radius="0.0174"></sphere>
625         </shape>
626       </transform>
627
628       <transform id="m6" DEF="m6" translation='0 0 .3628'>
629         <shape>
630           <appearance>
631             <material diffuseColor='0 1 0'></material>
632           </appearance>
633
634           <sphere radius="0.0174"></sphere>
635         </shape>
636       </transform>
637       <transform id="m7" DEF="m7" translation='0 0 .3698'>
638         <shape>
639           <appearance>
640             <material diffuseColor='0 1 0'></material>
641           </appearance>
642
643           <sphere radius="0.0174"></sphere>
644         </shape>
645       </transform>
646
647       <transform id="m8" DEF="m8" translation='0 0 .3768'>
648         <shape>
649           <appearance>
650             <material diffuseColor='0 1 0'></material>
651           </appearance>
652
653           <sphere radius="0.0174"></sphere>
654         </shape>
655       </transform>
656
657       <transform id="m9" DEF="m9" translation='0 0 .3838'>
658         <shape>
659           <appearance>
```

```
660             <material diffuseColor='0 1 0'></material>
661           </appearance>
662
663           <sphere radius="0.0174"></sphere>
664         </shape>
665       </transform>
666
667       <transform id="m10" DEF="m10" translation='0 0 .3908'>
668         <shape>
669           <appearance>
670             <material diffuseColor='0 1 0'></material>
671           </appearance>
672
673           <sphere radius="0.0174"></sphere>
674         </shape>
675       </transform>
676
677       <transform id="m11" DEF="m11" translation='0 0 .3977'>
678         <shape>
679           <appearance>
680             <material diffuseColor='0 1 0'></material>
681           </appearance>
682
683           <sphere radius="0.0174"></sphere>
684         </shape>
685       </transform>
686       <transform id="m12" DEF="m12" translation='0 0 .4047'>
687         <shape>
688           <appearance>
689             <material diffuseColor='0 1 0'></material>
690           </appearance>
691
692           <sphere radius="0.0174"></sphere>
693         </shape>
694       </transform>
695
696       <transform id="m13" DEF="m13" translation='0 0 .4117'>
697         <shape>
698           <appearance>
```

```
699             <material diffuseColor='0 1 0'></material>
700           </appearance>
701
702           <sphere radius="0.0174"></sphere>
703         </shape>
704       </transform>
705
706       <transform id="m14" DEF="m14" translation='0 0 .4187'>
707         <shape>
708           <appearance>
709             <material diffuseColor='0 1 0'></material>
710           </appearance>
711
712           <sphere radius="0.0174"></sphere>
713         </shape>
714       </transform>
715       <transform id="m15" DEF="m15" translation='0 0 .4256'>
716         <shape>
717           <appearance>
718             <material diffuseColor='0 1 0'></material>
719           </appearance>
720
721           <sphere radius="0.0174"></sphere>
722         </shape>
723       </transform>
724
725       <transform id="m16" DEF="m16" translation='0 0 .4326'>
726         <shape>
727           <appearance>
728             <material diffuseColor='0 1 0'></material>
729           </appearance>
730
731           <sphere radius="0.0174"></sphere>
732         </shape>
733       </transform>
734
735       <transform id="m17" DEF="m17" translation='0 0 .4396'>
736         <shape>
737           <appearance>
```

```
738             <material diffuseColor='0 1 0'></material>
739           </appearance>
740
741           <sphere radius="0.0174"></sphere>
742         </shape>
743       </transform>
744       <transform id="m18" DEF="m18" translation='0 0 .4466'>
745         <shape>
746           <appearance>
747             <material diffuseColor='0 1 0'></material>
748           </appearance>
749
750           <sphere radius="0.0174"></sphere>
751         </shape>
752       </transform>
753
754       <transform id="m19" DEF="m19" translation='0 0 .4536'>
755         <shape>
756           <appearance>
757             <material diffuseColor='0 1 0'></material>
758           </appearance>
759
760           <sphere radius="0.0174"></sphere>
761         </shape>
762       </transform>
763
764       <transform id="m20" DEF="m20" translation='0 0 .4605'>
765         <shape>
766           <appearance>
767             <material diffuseColor='0 1 0'></material>
768           </appearance>
769
770           <sphere radius="0.0174"></sphere>
771         </shape>
772       </transform>
773
774       <transform id="m21" DEF="m21" translation='0 0 .4675'>
775         <shape>
776           <appearance>
```

```
777             <material diffuseColor='0 1 0'></material>
778           </appearance>
779
780           <sphere radius="0.0174"></sphere>
781         </shape>
782       </transform>
783       <transform id="m22" DEF="m22" translation='0 0 .4745'>
784         <shape>
785           <appearance>
786             <material diffuseColor='0 1 0'></material>
787           </appearance>
788
789           <sphere radius="0.0174"></sphere>
790         </shape>
791       </transform>
792
793       <transform id="m23" DEF="m23" translation='0 0 .4815'>
794         <shape>
795           <appearance>
796             <material diffuseColor='0 1 0'></material>
797           </appearance>
798
799           <sphere radius="0.0174"></sphere>
800         </shape>
801       </transform>
802
803       <transform id="m24" DEF="m24" translation='0 0 .4884'>
804         <shape>
805           <appearance>
806             <material diffuseColor='0 1 0'></material>
807           </appearance>
808
809           <sphere radius="0.0174"></sphere>
810         </shape>
811       </transform>
812
813       <transform id="m25" DEF="m25" translation='0 0 .4954'>
814         <shape>
815           <appearance>
```

```
            <material diffuseColor='0 1 0'></material>
          </appearance>

          <sphere radius="0.0174"></sphere>
        </shape>
      </transform>
      <transform id="m26" DEF="m26" translation='0 0 .5024'>
        <shape>
          <appearance>
            <material diffuseColor='0 1 0'></material>
          </appearance>

          <sphere radius="0.0174"></sphere>
        </shape>
      </transform>

      <transform id="m27" DEF="m27" translation='0 0 .5094'>
        <shape>
          <appearance>
            <material diffuseColor='0 1 0'></material>
          </appearance>

          <sphere radius="0.0174"></sphere>
        </shape>
      </transform>

      <transform id="m28" DEF="m28" translation='0 0 .5164'>
        <shape>
          <appearance>
            <material diffuseColor='0 1 0'></material>
          </appearance>

          <sphere radius="0.0174"></sphere>
        </shape>
      </transform>
      <transform id="m29" DEF="m29" translation='0 0 .5233'>
        <shape>
          <appearance>
            <material diffuseColor='0 1 0'></material>
```

```
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="m30" DEF="m30" translation='0 0 .5303'>
    <shape>
      <appearance>
        <material diffuseColor='0 1 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="m31" DEF="m31" translation='0 0 .5373'>
    <shape>
      <appearance>
        <material diffuseColor='0 1 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>
  <transform id="m32" DEF="m32" translation='0 0 .5443'>
    <shape>
      <appearance>
        <material diffuseColor='0 1 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="m33" DEF="m33" translation='0 0 .5512'>
    <shape>
      <appearance>
        <material diffuseColor='0 1 0'></material>
```

```
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="m34" DEF="m0" translation='0 0 .5582'>
    <shape>
      <appearance>
        <material diffuseColor='0 1 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>
  <transform id="m35" DEF="m1" translation='0 0 .5652'>
    <shape>
      <appearance>
        <material diffuseColor='0 1 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="m36" DEF="m2" translation='0 0 .5722'>
    <shape>
      <appearance>
        <material diffuseColor='0 1 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="m37" DEF="m3" translation='0 0 .5792'>
    <shape>
      <appearance>
        <material diffuseColor='0 1 0'></material>
```

```
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>
  <transform id="m38" DEF="m4" translation='0 0 .5861'>
    <shape>
      <appearance>
        <material diffuseColor='0 1 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="m39" DEF="m5" translation='0 0 .5931'>
    <shape>
      <appearance>
        <material diffuseColor='0 1 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="m40" DEF="m6" translation='0 0 .6001'>
    <shape>
      <appearance>
        <material diffuseColor='0 1 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>
  <transform id="m41" DEF="m7" translation='0 0 .6071'>
    <shape>
      <appearance>
        <material diffuseColor='0 1 0'></material>
      </appearance>
```

```
    <sphere radius="0.0174"></sphere>
  </shape>
</transform>

<transform id="m42" DEF="m8" translation='0 0 .6140'>
  <shape>
    <appearance>
      <material diffuseColor='0 1 0'></material>
    </appearance>

    <sphere radius="0.0174"></sphere>
  </shape>
</transform>

<transform id="m43" DEF="m1" translation='0 0 .6210'>
  <shape>
    <appearance>
      <material diffuseColor='0 1 0'></material>
    </appearance>

    <sphere radius="0.0174"></sphere>
  </shape>
</transform>

<transform id="m44" DEF="m1" translation='0 0 .6280'>
  <shape>
    <appearance>
      <material diffuseColor='0 1 0'></material>
    </appearance>

    <sphere radius="0.0174"></sphere>
  </shape>
</transform>

    <!--Tip Section Nodes-->

<transform id="t0" DEF="t0" translation='0 0 .6350'>
  <shape>
```

```
      <appearance>
        <material diffuseColor='1 0 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>
  <transform id="t1" DEF="t1" translation='0 0 .6420'>
    <shape>
      <appearance>
        <material diffuseColor='1 0 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="t2" DEF="t2" translation='0 0 .6489'>
    <shape>
      <appearance>
        <material diffuseColor='1 0 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="t3" DEF="t3" translation='0 0 .6559'>
    <shape>
      <appearance>
        <material diffuseColor='1 0 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>
  <transform id="t4" DEF="t4" translation='0 0 .6629'>
    <shape>
      <appearance>
```

```
      <material diffuseColor='1 0 0'></material>
    </appearance>

    <sphere radius="0.0174"></sphere>
  </shape>
</transform>

<transform id="t5" DEF="t5" translation='0 0 .6699'>
  <shape>
    <appearance>
      <material diffuseColor='1 0 0'></material>
    </appearance>

    <sphere radius="0.0174"></sphere>
  </shape>
</transform>

<transform id="t6" DEF="t6" translation='0 0 .6768'>
  <shape>
    <appearance>
      <material diffuseColor='1 0 0'></material>
    </appearance>

    <sphere radius="0.0174"></sphere>
  </shape>
</transform>
<transform id="t7" DEF="t7" translation='0 0 .6838'>
  <shape>
    <appearance>
      <material diffuseColor='1 0 0'></material>
    </appearance>

    <sphere radius="0.0174"></sphere>
  </shape>
</transform>

<transform id="t8" DEF="t8" translation='0 0 .6908'>
  <shape>
    <appearance>
```

```
      <material diffuseColor='1 0 0'></material>
    </appearance>

    <sphere radius="0.0174"></sphere>
  </shape>
</transform>

<transform id="t9" DEF="t0" translation='0 0 .6978'>
  <shape>
    <appearance>
      <material diffuseColor='1 0 0'></material>
    </appearance>

    <sphere radius="0.0174"></sphere>
  </shape>
</transform>

<transform id="t10" DEF="t2" translation='0 0 .7048'>
  <shape>
    <appearance>
      <material diffuseColor='1 0 0'></material>
    </appearance>

    <sphere radius="0.0174"></sphere>
  </shape>
</transform>

<transform id="t11" DEF="t3" translation='0 0 .7117'>
  <shape>
    <appearance>
      <material diffuseColor='1 0 0'></material>
    </appearance>

    <sphere radius="0.0174"></sphere>
  </shape>
</transform>
<transform id="t12" DEF="t4" translation='0 0 .7187'>
  <shape>
    <appearance>
```

```
        <material diffuseColor='1 0 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="t13" DEF="t5" translation='0 0 .7257'>
    <shape>
      <appearance>
        <material diffuseColor='1 0 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="t14" DEF="t6" translation='0 0 .7327'>
    <shape>
      <appearance>
        <material diffuseColor='1 0 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>
  <transform id="t15" DEF="t7" translation='0 0 .7396'>
    <shape>
      <appearance>
        <material diffuseColor='1 0 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="t16" DEF="t8" translation='0 0 .7466'>
    <shape>
      <appearance>
```

```
      <material diffuseColor='1 0 0'></material>
    </appearance>

    <sphere radius="0.0174"></sphere>
  </shape>
</transform>

<transform id="t17" DEF="t0" translation='0 0 .7536'>
  <shape>
    <appearance>
      <material diffuseColor='1 0 0'></material>
    </appearance>

    <sphere radius="0.0174"></sphere>
  </shape>
</transform>
<transform id="t18" DEF="t1" translation='0 0 .7606'>
  <shape>
    <appearance>
      <material diffuseColor='1 0 0'></material>
    </appearance>

    <sphere radius="0.0174"></sphere>
  </shape>
</transform>

<transform id="t19" DEF="t1" translation='0 0 .7676'>
  <shape>
    <appearance>
      <material diffuseColor='1 0 0'></material>
    </appearance>

    <sphere radius="0.0174"></sphere>
  </shape>
</transform>

<transform id="t20" DEF="t2" translation='0 0 .7745'>
  <shape>
    <appearance>
```

```
        <material diffuseColor='1 0 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="t21" DEF="t3" translation='0 0 .7815'>
    <shape>
      <appearance>
        <material diffuseColor='1 0 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>
  <transform id="t22" DEF="t4" translation='0 0 .7885'>
    <shape>
      <appearance>
        <material diffuseColor='1 0 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="t23" DEF="t5" translation='0 0 .7955'>
    <shape>
      <appearance>
        <material diffuseColor='1 0 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="t24" DEF="t8" translation='0 0 .8024'>
    <shape>
      <appearance>
```

```
        <material diffuseColor='1 0 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="t25" DEF="t0" translation='0 0 .8094'>
    <shape>
      <appearance>
        <material diffuseColor='1 0 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>
  <transform id="t26" DEF="t1" translation='0 0 .8164'>
    <shape>
      <appearance>
        <material diffuseColor='1 0 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="t27" DEF="t2" translation='0 0 .8234'>
    <shape>
      <appearance>
        <material diffuseColor='1 0 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="t28" DEF="t3" translation='0 0 .8304'>
    <shape>
      <appearance>
```

```
            <material diffuseColor='1 0 0'></material>
          </appearance>

          <sphere radius="0.0174"></sphere>
        </shape>
      </transform>
      <transform id="t29" DEF="t4" translation='0 0 .8373'>
        <shape>
          <appearance>
            <material diffuseColor='1 0 0'></material>
          </appearance>

          <sphere radius="0.0174"></sphere>
        </shape>
      </transform>

      <transform id="t30" DEF="t5" translation='0 0 .8443'>
        <shape>
          <appearance>
            <material diffuseColor='1 0 0'></material>
          </appearance>

          <sphere radius="0.0174"></sphere>
        </shape>
      </transform>

      <transform id="t31" DEF="t6" translation='0 0 .8513'>
        <shape>
          <appearance>
            <material diffuseColor='1 0 0'></material>
          </appearance>

          <sphere radius="0.0174"></sphere>
        </shape>
      </transform>
      <transform id="t32" DEF="t7" translation='0 0 .8583'>
        <shape>
          <appearance>
            <material diffuseColor='1 0 0'></material>
```

```
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="t33" DEF="t8" translation='0 0 .8652'>
    <shape>
      <appearance>
        <material diffuseColor='1 0 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="t34" DEF="t0" translation='0 0 .8722'>
    <shape>
      <appearance>
        <material diffuseColor='1 0 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>
  <transform id="t35" DEF="t1" translation='0 0 .8792'>
    <shape>
      <appearance>
        <material diffuseColor='1 0 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="t36" DEF="t2" translation='0 0 .8862'>
    <shape>
      <appearance>
        <material diffuseColor='1 0 0'></material>
```

```
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="t37" DEF="t3" translation='0 0 .8932'>
    <shape>
      <appearance>
        <material diffuseColor='1 0 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>
  <transform id="t38" DEF="t4" translation='0 0 .9001'>
    <shape>
      <appearance>
        <material diffuseColor='1 0 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="t39" DEF="t5" translation='0 0 .9071'>
    <shape>
      <appearance>
        <material diffuseColor='1 0 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="t40" DEF="t6" translation='0 0 .9141'>
    <shape>
      <appearance>
        <material diffuseColor='1 0 0'></material>
```

```
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>
  <transform id="t41" DEF="t7" translation='0 0 .9211'>
    <shape>
      <appearance>
        <material diffuseColor='1 0 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="t42" DEF="t8" translation='0 0 .9280'>
    <shape>
      <appearance>
        <material diffuseColor='1 0 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="t43" DEF="t1" translation='0 0 .9350'>
    <shape>
      <appearance>
        <material diffuseColor='1 0 0'></material>
      </appearance>

      <sphere radius="0.0174"></sphere>
    </shape>
  </transform>

  <transform id="t44" DEF="t1" translation='0 0 .9420'>
    <shape>
      <appearance>
        <material diffuseColor='1 0 0'></material>
```

```
1440          </appearance>
1441
1442          <sphere radius="0.0174"></sphere>
1443        </shape>
1444      </transform>
1445
1446    </scene>
1447  </x3d>
1448  </div>
1449
1450 </body>
1451 <footer>
1452
1453 </footer>
1454
1455 </html>
```

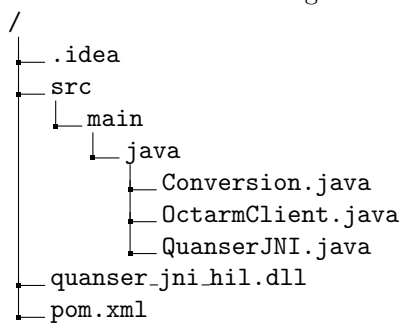
Appendix L server.js

```
var express = require('express');
var app = express();
var server = require('http').Server(app);
var io = require('socket.io')(server);

app.set('port', (process.env.PORT || 3000));
app.use(express.static(__dirname + '/public'));
app.set('views', __dirname+'/views');
app.set('view engine', 'ejs');

app.get('/', function(req, res){
  res.render('model');
});

io.on('connection', function(socket){

  console.log('a user connected');
  socket.on('disconnect', function(){
    console.log('user disconnected');
  });


  socket.on('message', function(msg){
    console.log("Wrote Data\n");
    io.emit('message', msg);
  });
});

server.listen(3000, function(){
  console.log('listening on port: ', app.get('port'));
});
```

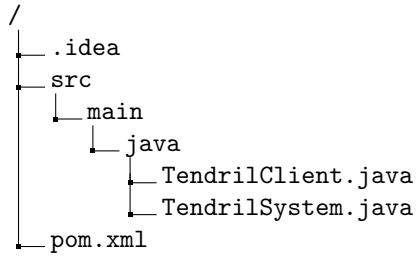
Appendix C

Client Code File Listing

Octarm Code Listing



Tendril Code Listing



Appendix A Conversion.java

```
public class Conversion {

    // INPUTS
    // Base: s1, k1, phi1
    // Mid: s2, k2, phi2
    // Tip: s3, k3, phi3

    //// OUTPUTS
    // Base: l1b, l2b, l3b
    // Mid: l1m, l2m, l3m
    // Tip: l1t, l2t, l3t

    //// CONSTANTS USED
    // Base
    float pi = (float) Math.PI;
    float db = 27.5f / (2 * pi);
    // Mid
    float dm = 27.5f / (2 * pi);
    // Tip
    float dt = 23.58f / (2 * pi);

    int n = 7;

    public float[] convert(float[] params) {

        float s1 = params[0];
        float k1 = params[1]*0.0254f;
        float phi1 = params[2];
        float s2 = params[3];
        float k2 = params[4]*0.0254f;
        float phi2 = params[5];
```

```
float s3 = params[6];
float k3 = params[7]*0.0254f;
float phi3 = params[8];

//// S K PHI TO L1 L2 L3 CONVERSION

// Base
float l1b = 2 * n * (float)Math.sin(s1 * k1 / (2 * n)) * ((1 / k1) - (db *
    (float)Math.sin(phi1)));
float l2b = 2 * n * (float)Math.sin(s1 * k1 / (2 * n)) * ((1 / k1) + (db *
    (float)Math.sin((pi / 3) + phi1)));
float l3b = 2 * n * (float)Math.sin(s1 * k1 / (2 * n)) * ((1 / k1) - (db *
    (float)Math.cos((pi / 6) + phi1)));

// Mid
float l1m = 2 * n * (float)Math.sin(s2 * k2 / (2 * n)) * ((1 / k2) - (dm *
    (float)Math.sin(phi2)));
float l2m = 2 * n * (float)Math.sin(s2 * k2 / (2 * n)) * ((1 / k2) + (dm *
    (float)Math.sin((pi / 3) + phi2)));
float l3m = 2 * n * (float)Math.sin(s2 * k2 / (2 * n)) * ((1 / k2) - (dm *
    (float)Math.cos((pi / 6) + phi2)));

// Tip

float l1t = 2 * n * (float)Math.sin(s3 * k3 / (2 * n)) * ((1 / k3) - (dt *
    (float)Math.sin(phi3)));
float l2t = 2 * n * (float)Math.sin(s3 * k3 / (2 * n)) * ((1 / k3) + (dt *
    (float)Math.sin((pi / 3) + phi3)));
float l3t = 2 * n * (float)Math.sin(s3 * k3 / (2 * n)) * ((1 / k3) - (dt *
    (float)Math.cos((pi / 6) + phi3)));

//// L1 L2 L3 TO PRESSURE REGULATOR GAIN CONVERSION
```

```
        float sc1t = 49.312f * l1t - 16.492f;
        float  sc2t = 46.165f * l2t - 15.156f;
        float sc3t = 52.034f * l3t - 17.657f;

        // Mid
        float sc1m = 52.828f * l1m - 16.131f;
        float sc2m = 58.287f * l2m - 18.091f;
        float sc3m = 57.692f * l3m - 18.099f;

        // Base
        float sc1b = 60.046f * l1b - 19.294f;
        float sc2b = 60.514f * l2b - 19.529f;
        float sc3b = 59.612f * l3b - 18.81f;

        float[] voltages = {sc1b, sc3b, sc2b, sc1m, sc3m, sc2m, sc1t, sc3t, sc2t };
        return voltages;
    }
}
```

Appendix B OctarmClient.java

```
import io.socket.client.IO;
import io.socket.client.Socket;
import io.socket.emitter.Emitter;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.net.URISyntaxException;
import java.util.concurrent.CountDownLatch;


public class OctarmClient {

    static Socket socket;

    private final CountDownLatch exitLatch;

    public static void main(String[] args){
        System.out.println("Java Client Started");
        try{
            new OctarmClient();
        } catch (Exception e){
            e.printStackTrace();
        }

    }

    public OctarmClient() throws Exception{
        setupConnection();
        socket.connect();
```

```
    while (!socket.connected()){ }


    exitLatch = new CountDownLatch(1);

    try{
        exitLatch.await();
    } catch (InterruptedException e){
        e.printStackTrace();
    }

    System.out.println("shutdown");
    socket.disconnect();
    while (socket.connected()){}
    socket.off();


}

public void setupConnection() throws URISyntaxException {


    socket = IO.socket("http://localhost:3000");
    System.out.println("setup connection");

    socket.on(Socket.EVENT_CONNECT, new Emitter.Listener() {
        public void call(Object... objects) {
            socket.emit("connection", "hi");
            System.out.println("Socket Connected");

        }
    });
```

```
socket.on(Socket.EVENT_DISCONNECT, new Emitter.Listener() {
    public void call(Object... objects) {
        System.out.println("Socket Disconnected");
    }
});

socket.on(Socket.EVENT_ERROR, new Emitter.Listener() {
    public void call(Object... objects) {
        System.out.println("socket error");
    }
});

socket.on("message", new Emitter.Listener() {
    public void call(Object... objects) {
        try{
            JSONObject jsonData = new JSONObject(objects[0].toString());
            JSONArray paramArray = jsonData.getJSONArray("params");
            float[] paramData = new float[9];
            if(paramArray !=null) {
                for (int i = 0; i < paramArray.length(); i++) {
                    JSONObject param = paramArray.getJSONObject(i);
                    paramData[i] =
                        Float.parseFloat(param.getString(JSONObject.getNames(param)[0]));
                }

                float[] vals = new Conversion().convert(paramData);
                for (int i = 0; i < vals.length; i++) {
                    System.out.println(vals[i]);
                }
                Quanser Q8 = new Quanser();
                Q8.connectWrite(vals);

            }
```

```
                } catch (JSONException e){
                    e.printStackTrace();
                }

            }
        });
    }

}
```

Appendix C QuanserJNI.java

```
public class Quanser{
  static{
    System.loadLibrary("quanser_jni_hil");
  }
  public native void connectWrite(float[] voltage);
}
```

Appendix D quanser_jni_hil.c

```
#include <jni.h>
#include <stdio.h>
#include <hil.h>
#include <quanser_messages.h>
#include "quanser_jni_hil.h"

JNIEXPORT void JNICALL Java_Quanser_connectWrite(JNIEnv *env, jobject thisObj, jfloatArray
    voltage){
  t_card board;
  t_card mini_board;

  int i;
  t_uint32 channels[] = {0,1,2,3,4,5,6,7};
  t_uint32 minichannel[] = {0};
  jfloat *fltArray = (*env)->GetFloatArrayElements(env,voltage,0);

  t_double buffer[9];
  t_double buffer2[] = {fltArray[8]};

  t_error write_result;

  t_error result = hil_open("q8_usb", "0", &board);
  t_error res2 = hil_open("q8_usb", "1", &mini_board);

  for(i=0; i<9; i++){
     buffer[i] = fltArray[i];
  }

  if(result == 0){
     printf("connected to quanser board");
     write_result = hil_write_analog(board, channels, ARRAY_LENGTH(channels), buffer);
```

```
      write_result = hil_write_analog(mini_board, minichannel, ARRAY_LENGTH(minichannel),
          buffer2);
      hil_close(board);
      hil_close(mini_board);

   } else {
      printf("error");
   }

   return;
}
```

Appendix E quanser_jni_hil.h

```
/* DO NOT EDIT THIS FILE - it is machine generated */
#include <jni.h>
/* Header for class Quanser */

#ifndef _Included_QuanserJNI
#define _Included_QuanserJNI
#ifdef __cplusplus
extern "C" {
#endif
/*
 * Class:    QuanserJNI
 * Method:   connectWrite
 * Signature: (I)V
 */
JNIEXPORT void JNICALL Java_QuanserJNI_connectWrite
  (JNIEnv *, jobject, jfloatArray);

#ifdef __cplusplus
}
#endif
#endif
```

Appendix F TendrilClient.java

```
import io.socket.client.IO;
import io.socket.client.Socket;
import io.socket.emitter.Emitter;
import jssc.SerialPort;
import jssc.SerialPortEvent;
import jssc.SerialPortEventListener;
import jssc.SerialPortException;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;



import java.net.URISyntaxException;
import java.util.concurrent.CountDownLatch;

public class TendrilClient {

    public static Socket socket;
    private static final String PORT = "COM4";
    private static final int BAUD_RATE = 9600;
    private static SerialPort serialPort;
    private final CountDownLatch exitLatch;
    private float[] defaults = {0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f};
    public TendrilSystem previousSystem;



    public static void main(String[] args ) {


        System.out.println("Java Tendril Client Started");
        try {
            new TendrilClient();
```

```
    } catch (Exception e) {
        e.printStackTrace();
    }
}



public TendrilClient() throws Exception{
    previousSystem = new TendrilSystem(defaults);
    setupConnection();
    socket.connect();
    serialPort = new SerialPort(PORT);
    configureSerialPort();

    while(!socket.connected()){ }

    exitLatch = new CountDownLatch(1);

    try {
        exitLatch.await();
    } catch (InterruptedException e){
        e.printStackTrace();
    }
}

private static class PortReader implements SerialPortEventListener {
    //@Override
    public void serialEvent(SerialPortEvent event){
        if(event.isRXCHAR() && event.getEventValue() > 0){
            try {

                String ReceiveData = serialPort.readString(event.getEventValue());
                System.out.println(ReceiveData);
```

```
                System.out.println("");
            } catch (SerialPortException e){
                e.printStackTrace();
            }
        }
    }
}

public void configureSerialPort(){
    try{
        System.out.println("Port opened: " + serialPort.openPort());
        serialPort.setParams(SerialPort.BAUDRATE_9600,
                SerialPort.DATABITS_8,
                SerialPort.STOPBITS_1,
                SerialPort.PARITY_NONE);

        serialPort.addEventListener(new PortReader());

    } catch (SerialPortException e){
        e.printStackTrace();
    }
}

public void setupConnection() throws URISyntaxException {

    socket = IO.socket("http://localhost:3000");
    System.out.println("setup connection");

    socket.on(Socket.EVENT_CONNECT, new Emitter.Listener() {
        public void call(Object... objects) {
            socket.emit("connection", "hi");
            System.out.println("Socket Connected");
```

```
    }
});

socket.on(Socket.EVENT_DISCONNECT, new Emitter.Listener() {
    public void call(Object... objects) {
        System.out.println("Socket Disconnected");
    }
});

socket.on(Socket.EVENT_ERROR, new Emitter.Listener() {
    public void call(Object... objects) {
        System.out.println("socket error");
    }
});

socket.on("message", new Emitter.Listener() {
    public void call(Object... objects) {
        try{
            JSONObject jsonData = new JSONObject(objects[0].toString());
            JSONArray paramArray = jsonData.getJSONArray("params");
            float[] paramData = new float[9];
            if(paramArray !=null){
                for (int i = 0; i < paramArray.length(); i++) {
                    JSONObject param = paramArray.getJSONObject(i);
                    paramData[i] =
                        Float.parseFloat(param.getString(JSONObject.getNames(param)[0]));

                }

                TendrilSystem currentSystem = new TendrilSystem(paramData);

                System.out.println("Tension: "+currentSystem.baseTension+"%
                    "+currentSystem.midTension+"% "+currentSystem.distalTension+"%");
```

```
                    currentSystem.findDistalTensions();
                    System.out.println("");

                    if(currentSystem.changedSystemStatus(previousSystem, serialPort)){
                        previousSystem = new TendrilSystem(paramData);
                    }
                }
            } catch (JSONException e){
                e.printStackTrace();
            }
        }
    });
  }
}
```

Appendix G TendrilSystem.java

```
import jssc.SerialPort;
import jssc.SerialPortException;

import java.nio.ByteBuffer;
import java.util.ArrayList;
import java.util.Vector;

public class TendrilSystem {

    private float baseMaxK = 1.25f;
    private float midMaxK = 1.25f;
    private float distalMaxK = 3.5f;
    private float maxPhi = 360f;
    private float[] motorLocations = {0, 120, 240, 40, 160, 280, 80, 200, 320};
    private byte[] motorTensions = {0, 0,  0,  0,   0,   0,  0,   0,   0};
    private float base_k;
    private float base_phi;
    private float mid_k;
    private float mid_phi;
    private float distal_k;
    private float distal_phi;
    public byte baseTension;
    public byte midTension;
    public byte distalTension;


    public TendrilSystem(float[] params) {
        this.base_k = params[1];
        this.base_phi = params[2] * 180 / (float) Math.PI;
        this.mid_k = params[4];
```

```
        this.mid_phi = params[5] * 180 / (float) Math.PI;
        this.distal_k = params[7];
        this.distal_phi = params[8] * 180 / (float) Math.PI;

        if (base_phi > maxPhi) {
            base_phi -= maxPhi;
        }

        if (mid_phi > maxPhi) {
            mid_phi -= maxPhi;
        }

        if (distal_phi > maxPhi) {
            distal_phi -= maxPhi;
        }

        baseTension = convertToByte(baseTensionPercentage(base_k));
        midTension = convertToByte(midTensionPercentage(mid_k));
        distalTension = convertToByte(distalTensionPercentage(distal_k));
        findBaseTensions();
        findMidTensions();
        findDistalTensions();

    }

    public boolean changedSystemStatus(TendrilSystem prev, SerialPort sp){
        ArrayList<Byte> bytes = new ArrayList<Byte>();
        for (int i = 0; i < motorTensions.length ; i++) {
            if (byteSubtract(this.motorTensions[i],prev.motorTensions[i]) >= 15) {
                bytes.add(convertToByte(i+1));
                bytes.add(convertToByte(this.motorTensions[i]+1));
            }
        }
```

```
    if (bytes.size() > 0){
        byte[] buffer = new byte[bytes.size()];
        System.out.println("Writing...");
        for (int i = 0; i <bytes.size() ; i++) {
            buffer[i] = bytes.get(i);
            System.out.println("buffer["+i+"}: "+ buffer[i]);
        }
        try{
            sp.writeBytes(buffer);
        } catch (SerialPortException e){
            e.printStackTrace();
        }
        return true;
    } else{
        return false;
    }
}

private int byteSubtract(byte a, byte b){
    int num1 = a;
    int num2 = b;
    return num1 - num2;
}



public void findBaseTensions(){
    if(base_phi >= 0 && base_phi <= 120){
        //between 0 and 1
        motorTensions[2] = (byte)0;
        float ratio = (motorLocations[1]-base_phi)/120;
        motorTensions[0] = convertToByte(Math.round(baseTension*ratio));
        motorTensions[1] = convertToByte(Math.round(baseTension*(1-ratio)));
    } else if (base_phi >= 120 && base_phi <= 240){
```

```
        //between 1 and 2
        motorTensions[0] = (byte)0;
        float ratio = (motorLocations[2]-base_phi)/120;
        motorTensions[1] = convertToByte(Math.round(baseTension*ratio));
        motorTensions[2] = convertToByte(Math.round(baseTension*(1-ratio)));
    } else{
        //between 2 and 0
        motorTensions[1] = (byte)0;
        float ratio = (360-base_phi)/120;
        motorTensions[2] = convertToByte(Math.round(baseTension*ratio));
        motorTensions[0] = convertToByte(Math.round(baseTension*(1-ratio)));
    }

    System.out.println("0: "+ motorTensions[0]+" 1: "+motorTensions[1]+" 2:
         "+motorTensions[2]);
}

public void findMidTensions(){
    if(mid_phi >= 40 && mid_phi <= 160){
        //between 3 and 4
        motorTensions[5] = (byte)0;
        float ratio = (motorLocations[4]-mid_phi)/120;
        motorTensions[3] = convertToByte(Math.round(midTension*ratio));
        motorTensions[4] = convertToByte(Math.round(midTension*(1-ratio)));
    } else if (mid_phi >= 160 && mid_phi <= 280){
        //between 4 and 5
        motorTensions[3] = (byte)0;
        float ratio = (motorLocations[5]-mid_phi)/120;
        motorTensions[4] = convertToByte(Math.round(midTension*ratio));
        motorTensions[5] = convertToByte(Math.round(midTension*(1-ratio)));
    } else{
        //between 5 and 3
        motorTensions[4] = (byte)0;
```

```
        if(mid_phi <=40) mid_phi+=360;
        float ratio = (400-mid_phi)/120;
        motorTensions[5] = convertToByte(Math.round(midTension*ratio));
        motorTensions[3] = convertToByte(Math.round(midTension*(1-ratio)));
    }

    System.out.println("3: "+ motorTensions[3]+" 4: "+motorTensions[4]+" 5:
        "+motorTensions[5]);
}

public void findDistalTensions(){
    if(distal_phi >= 80 && distal_phi <= 200){
        //between 6 and 7
        motorTensions[8] = (byte)0;
        float ratio = (motorLocations[7]-distal_phi)/120;
        motorTensions[6] = convertToByte(Math.round(distalTension*ratio));
        motorTensions[7] = convertToByte(Math.round(distalTension*(1-ratio)));
    } else if (distal_phi >= 200 && distal_phi <= 320){
        //between 7 and 8
        motorTensions[6] = (byte)0;
        float ratio = (motorLocations[8]-distal_phi)/120;
        motorTensions[7] = convertToByte(Math.round(distalTension*ratio));
        motorTensions[8] = convertToByte(Math.round(distalTension*(1-ratio)));
    } else{
        //between 8 and 6
        motorTensions[7] = (byte)0;
        if(distal_phi <=80) distal_phi+=360;
        float ratio = (440-distal_phi)/120;
        motorTensions[8] = convertToByte(Math.round(distalTension*ratio));
        motorTensions[6] = convertToByte(Math.round(distalTension*(1-ratio)));
    }
```

```
        System.out.println("6: "+ motorTensions[6]+" 7: "+motorTensions[7]+" 8:
            "+motorTensions[8]);
    }

    public byte convertToByte(int in){

        return Byte.parseByte(Integer.toBinaryString(in),2);
    }

    public int baseTensionPercentage(float sec_k){
        return Math.round(100*sec_k/baseMaxK);
    }

    public int midTensionPercentage(float sec_k){
        return Math.round(100*sec_k/midMaxK);
    }

    public int distalTensionPercentage(float sec_k){
        return Math.round(100*sec_k/distalMaxK);
    }

}
```

Bibliography

[1] D. Brutzman and L. Daly. *Extensible 3D Graphics for Web Authors*. Morgan Kauffman, San Francisco, USA, 2007.

[2] James Coglan. Sylvester: Vector and matrix math for javascript. `sylvester.jcoglan.com`, 2016.

[3] Web3D Consortium. What is x3d. `http://www.web3d.org/x3d/what-x3d`, 2016.

[4] Microsoft Corporation. Visual studio homepage. `https://www.visualstudio.com/en-us/visual-studio-homepage-vs.aspx`, 2016.

[5] M. Csencsits, B.A. Jones, and W. McMahan. User interfaces for continuum robot arms. In *Proceedings IEEE/RSJ International Conference on Intelligent Robotic Systems*, pages 3011–3018, Edmonton, Canada, 2005.

[6] Jon Duckett. *JavaScript and JQuery: Interactive Front-End Web Development*. Wiley, Indianapolis, Indiana, 2014.

[7] ecma international. Introducing json. `http://json.org/`, 2016.

[8] Node.js Foundation. About node.js. `https://nodejs.org/en/about/`, 2016.

[9] The Apache Software Foundation. Apache maven project. `https://maven.apache.org/`, 2016.

[10] Fraunhofer-Gesselschaft. x3dom at a glance. `http://www.x3dom.org/`, 2016.

[11] C. Frazelle, A. Kapadia, K. Fry, and I. Walker. Teleoperation mappings from rigid link robots to their extensible continuum counterparts. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 1–7, Stockholm, Sweden, 2016.

[12] I.S. Godage, D.T. Branson, E. Guglielmino, G.A. Medrano-Cerda, and D.G. Caldwell. Shape function-based kinematics and dynamics for variable-length continuum robotic arms. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 452–457, Shanghai, China, 2011.

[13] I.S. Godage, E. Guglielmino, D.T. Branson, G.A. Medrano-Cerda, and D.G. Caldwell. Novel modal approach for kinematics of multisection continuum arms. In *Proceedings IEEE/RSJ International Conference on Intelligent Robotic Systems*, pages 1093–1098, San Francisco, California, 2011.

[14] G.Robinson and J.B.C. Davies. Continuum robots - a state of the art. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 2849–2854, Detroit, Michigan, 1999.

[15] A. Grzesiak, R. Becker, and A. Verl. The bionic handling assistant - a success story of additive manufacturing. *Assembly Automation*, 31(4):329–333, September 2011.

[16] E. Guglielmino, N. Tsagarakis, and D.G. Caldwell. An octopus-anatomy inspired robotics arm. In *Proceedings IEEE/RSJ International Conference on Intelligent Robotic Systems*, pages 3091–3096, Taipei, Taiwan, 2010.

[17] M.W. Hannan and I.D. Walker. Analysis and experiments with an elephant's trunk robot. *Advanced Robotics*, 15(8):847–858, August 2001.

[18] M.W. Hannan and I.D. Walker. Kinematics and the implementation of an elephant trunk manipulator and other continuum style robots. *Journal of Robotic Systems*, 20(2):45–63, February 2003.

[19] Ivor Horton. *Ivor Horton's Beginning Java*. Wiley, Indianapolis, Indiana, 2011.

[20] R. Webster III and B. A. Jones. Design and kinematic modeling of constant curvature continuum robots. *International Journal of Robotics Research*, 29(13):1661–1683, July 2010.

[21] J.Burgner-Kars, D.C. Rucker, and H. Choset. Continuum robots for medical applications: A survey. *IEEE Transactions on Robotics*, 31(6):1261–1280, December 2015.

[22] JetBrains. Intellij idea: Capable and ergonomic java ide. `https://www.jetbrains.com/idea/`, 2016.

[23] B. Jones and I. Walker. Three-dimensional modeling and display of continuum robots. In *Proceedings IEEE/RSJ International Conference on Intelligent Robotic Systems*, pages 5872–5877, Beijing, China, 2006.

[24] B. Jones and I.D. Walker. Kinematics of multisection continuum robots. *IEEE Transactions on Robotics*, 22(1):43–57, February 2006.

[25] The jQuery Foundation. What is jquery? `https://jquery.com/`, 2016.

[26] W. McMahan, M. Pritts, V. Chitrakaran, D. Dienno, M. Grissom, B. Jones, M. Csenscits, C.D. Rahn, D. Dawson, and I.D. Walker. Field trials and testing of octarm continuum robots. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 2336–2341, Orlando, Florida, 2006.

[27] Microsoft. Microsoft developer network msdn: Advantages of using dlls. `https://msdn.microsoft.com/en-us/library/dtba4t8b.aspx`, 2016.

[28] G. Niemeyer, C. Preusche, and G. Hirzinger. *Springer Handbook of Robotics*. Springer-Verlag, Berlin, Germany, 2008.

[29] Oracle. Java se documentation: Java native interface specification. `http://docs.oracle.com/javase/7/docs/technotes/guides/jni/spec/intro.html#wp725`, 2016.

[30] R.M.Scott and M.B. Wooten. Tendril validation: Hardware limits. `https://www.youtube.com/watch?v=vFJBHD8XOxY`, 2016.

[31] R.M. Scott. Octarm: Base section, midsection and tip section validation. `https://www.youtube.com/watch?v=pbrxQY4RZ1Q`, 2016.

[32] R.M. Scott. Octarm base section validation. `https://www.youtube.com/watch?v=dZFECFooogo`, 2016.

[33] R.M. Scott. Octarm midsection validation. https://www.youtube.com/watch?v=Ru51mc_Gl3o, 2016.

[34] R.M. Scott. Octarm tip section validation. https://www.youtube.com/watch?v=zHtm1-NxOjc, 2016.

[35] R.M. Scott and M.B. Wooten. Tendril validation: S bend. https://www.youtube.com/watch?v=oiLhKiW11XE, 2016.

[36] M.W. Spong, S. Hutchinson, and M. Vidyasagar. *Robotic Modeling and Control*. Wiley, Hoboken, NJ, 2006.

[37] B. Stanczyk and M. Buss. Development of a telerobotic system for exploration of hazardous environments. In *Proceedings IEEE/RSJ International Conference on Intelligent Robotic Systems*, pages 2532–2537, Sendai, Japan, 2004.

[38] Unity Technologies. Unity technologies company facts. https://unity3d.com/public-relations, 2016.

[39] D. Trivedi, C.D. Rahn, W.M. Kier, and I.D. Walker. Soft robotics: Biological inspiration, state of the art, and future research. *Applied Bionics and Biomechanics*, 5(3):99–117, September 2008.

[40] I.D. Walker. Continuous backbone "continuum" robot manipulators: A review. *ISRN Robotics*, 2013(1):1–19, July 2013.

[41] M.B. Wooten. Novel vine-like continuum robot for environmental exploration applications. Master's thesis, Clemson University, 2016.

[42] M.B. Wooten and I.D. Walker. A novel vine-like robot for in-orbit inspection. In *Proceedings 45th International Conference on Environmental Systems*, pages 1–11, Bellevue, Washington, 2015.